МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, доцент
_____ Т.С. Картбаев
«____» _____ 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

На тему: Модели и методы многоуровневой защиты информационных систем

Специальность 6М070400 – «Вычислительная техника и программное обеспечение»

Выполнил магистрант группы МВТн-16-2 _____ Жантлеуова А.К.

Научный руководитель д.т.н, профессор _____ Казиев Г.З.
«14» 06 2018 г.

Консультант по применению
вычислительной техники: ст. преп. _____ А.М. Рамазанова
«14» 06 2018 г.

Нормоконтролер: PhD, ст. преп. _____ Ж. Бидахмет
«06» 06 2018 г.

Рецензент: к.ф-м.н., асс. профессор (доцент) _____ Б.С. Каленова
«____» _____ 2018 г.

Алматы 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра IT-инжиниринг

Специальность 6М070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Жантлеуовой Асель Канатовне

Тема диссертации: Модели и методы многоуровневой защиты информационных систем

Утверждена приказом по университету № #10 от «08» 01 2018 г.

Срок сдачи законченной работы «____» _____ 2018 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта):Технический документ Microsoft об обеспечении безопасности данных SQL Server с помощью многоуровневой защитой.

Перечень вопросов, подлежащих разработке в диссертационной работе, или краткое содержание диссертационной работы:

а) общее понятие MLS;
б) анализ моделей и систем MLS;
в) общая модель для оценки многоуровневой безопасности объектно-ориентированных программ;
г) реализация многоуровневой модели разграничения доступа в базах данных SQL.

Перечень графического материала (с точным указанием обязательных чертежей): представлены 10 таблиц, 23 иллюстраций.

Основная рекомендуемая литература:

1 Anderson, J.P. (1972). Computer Security Technology Planning Study Volume II, ESD-TR-73-51.
2 Bell, D.D. and L.J. La Padula (1974). Secure Computer System: Unified Exposition and Multics Interpretation, ESD-TR-75-306. Bedford, MA: ESD/AFSC, Hanscom AFB.
3 Boehm, B.W. 1981, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall.

Консультации по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Программная часть	Рамазанова А.М.	20.05.2018-11.06.2018	
Нормоконтролер	Бидахмет Ж.	20.05.2018-06.06.2018	Ж. Бидахмет

График
подготовки дипломной работы

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Сбор информации по тематике MLS	20.11.2018	
Исследование и разработка	25.02.2018	
Получение результатов, документирование	25.05.2018	

Дата выдачи задания «__06__» _____10_____ 2017 г.

Заведующий кафедрой _____ Т.С. Картбаев

Научный руководитель работы _____ Г.З.Казиев

Задание принял к исполнению студент _____ А.К. Жантлеуова

Аңдатпа

Бұл диссертациялық жұмыста ақпараттық жүйелерді көп деңгейлі қорғау модельдері мен әдістері зерттелді. Объектілі-бағытталған бағдарламалардың көп деңгейлі қауіпсіздігін бағалау үшін жалпы үлгі ұсынылады. Осы модельге негізделген банк жүйесінің қауіпсіздік бөлігін бағалау жүргізілуде. SQL дерекқорларында кіруді басқарудың көп деңгейлі моделі іске асырылады.

Аннотация

В данной диссертационной работе исследуется модели и методы многоуровневой защиты информационных систем. Предлагаетсяобщая модель для оценки многоуровневой безопасности объектно-ориентированных программ. Проводится оценка части безопасности банковской системы на основе данной модели. РеализуетсямногоуровневаямодельразграничениядоступавбазахданныхSQL.

Annotation

In this dissertation work the models and methods of multi-level security of information systems are investigated. A general model for the assessment of multi-level security of object-oriented programs is proposed. An assessment of the security of the banking system on the basis of this model. A multilevel access control model is implemented in SQL databases.

Перечень сокращений

Сокращение	Пояснение
MLS	Multilevelsecurity
BLP	МодельБелл-ЛаПадула
MAC	Mandatoryaccesscontrol, мандатный контроль доступа
DAC	Discretionary access control, дискреционный контроль доступа
CCA	Chosen-ciphertext attack, атака с выбранным шифротекстом
ASCAM	Accessibility of Security-Critical Attributes Metric
ASCMM	Accessibility of Security-Critical Methods Metric
CSCMM	Cohesiveness of Security-Critical Methods Metric
CSCCM	Coupling of Security-Critical Classes Metric
SCDSM	Security-Critical Design Size Metric
БД	Базаданных
СУБД	Система управления базой данных

Введение

В двадцатом первом веке внедрение большого количества вычислительных технологий в структуры современных организаций стало объективной реальностью. На всех уровнях управления любой организации имеется желание расширить свои коммуникационные и информационные возможности путем внедрения современных возможностей. В результате это привело к тому, что информационные структуры организаций разрастаются: локальные сети организаций подключаются к интернету, использование больших данных, объединение в офисные многоуровневые структуры, разрастаются до уровня распределенных корпоративных сетей.

Внедрение средств вычислительной техники в структуру управления современных компаний и организаций является, бесспорно, прогрессивным процессом, поэтому вполне можно объяснить, что до определенного момента не возникает вопросов и опасений, связанных с использованием информационных технологий. Результатом этого часто является стихийный рост вычислительной инфраструктуры предприятия, которая по мере приобретения средств вычислительной техники разрастается, приобретая неструктурированный гетерогенный характер. Что в свою очередь приводит к неконтролируемому росту уязвимостей системы и увеличению возможностей доступа к секретной информации со стороны внешних и внутренних нарушителей. Таким образом, любая информационная система, используемая в организации, становится потенциально опасной для своих собственников, своего рода «миной замедленного действия», которая может запуститься в самый неожиданный момент.

Можно сказать, что безопасность – это один из наиболее важных и сложных аспектов построения и функционирования современных вычислительных информационных систем: от отдельно взятого приложения до сверхпроизводительного компьютерного кластера. Тема информационной безопасности охватывает весь жизненный цикл программного продукта: от зарождения идеи продукта до вывода системы ее из эксплуатации. В сфере информационных технологий под безопасностью обычно понимают защищенность информации (данных) и ее окружения от целенаправленных или случайных поступков, в результате которых может быть нанесен неприемлемый убыток самой информационной системе или ее пользователям.

Таким образом, при проектировании и дальнейшем эксплуатации информационных систем различного назначения возникают различные проблемы по обеспечениюих безопасности. И этоиграет ключевую роль. Данное заключение касается не только информационных вычислительных систем специального применения, но и систем общего назначения, используемых в обиходе.

Цель работы заключается в исследовании понятия «многоуровневой защиты информационных систем» (MLS) и реализация модели разграничения доступа.

Объектом исследования данной магистерской работы является информационные системы, а в точности часть банковской системы.

Предмет исследованиясоставляют модели и методы многоуровневой безопасности информационных систем, позволяющие провести оценку рисков состояния информационной системы.

Научная новизна заключалась в том, что были учтены и исправлены некоторые недостатки существующих советов компании Microsoft, то есть рекомендации по реализации MLSмодели доступа, а также представлена общая модель для оценки многоуровневой безопасности объектно-ориентированных программ.

Практическая новизна заключается в том, что данную работу, можно использовать как учебный материал в высших учебных заведениях и реализованную модель можно внедрить в информационную систему местного назначения.

По теме диссертации было опубликовано две статьи.

1Исследование понятия «многоуровневой защиты информационных систем»

1.1 Понятие «многоуровневая защита информационных систем (MLS)»

Многоуровневая защита информационных систем (MLS) представляет собой проблему для сообщества компьютерной безопасности с 1960-х годов. Многоуровневая защита звучит как обыденная проблема в управлении доступом: позволять свободно передавать информацию между получателями в вычислительной системе, которые имеют соответствующие разрешения безопасности, одновременно с этим предотвращая утечки несанкционированным пользователям. Однако системы многоуровневой защиты включают в себя две основные функции [1]:

- система должна применять ограничения независимо от действий системных пользователей или администраторов;
- системы MLS стремятся осуществить эти ограничения с невероятно высокой надежностью.

Это привело к тому, что разработчики внедрили специализированные механизмы защиты и применили сложные методы для проверки, анализа и тестирования этих механизмов для корректного и надежного поведения.

Многие предприятия и организации должны защищать секретную информацию и большинство из них могут допустить некоторую ее утечку. Однако организации, которые используют MLS, не допускают утечки вообще. Предприятия могут столкнуться с юридическими или финансовыми рисками, если они не в состоянии защитить коммерческую тайну, но они могут, как правило, восстановиться впоследствии, возместив убыток. В худшем случае бизнес обанкротится. Менеджеры, которые рискуют бизнес секретами, могут потерять свои рабочие места, если секреты просочатся, но они, скорее всего, потеряют свои рабочие места из-за неудачных проектов или переполненных бюджетов. Это накладывает ограничение на сумму денег, которую бизнес будет вкладывать в секретность данных.

Оборонное сообщество, которое включает в себя военные службы, разведывательные организации, связанные с ними государственные учреждения и поддерживающие их предприятия, не может легко оправиться от определенных утечек информации. Системы скрытности не являются скрытными, если цели знают, что искать, и системы наблюдения не видят вещей, если цели знают, какой камуфляж использовать. Такие неудачи не всегда можно исправить, просто потратив больше денег. Еще хуже то, что слабость системы может быть обнаружена только после того, как дипломатическая или военная катастрофа уже произойдет. Во время «холодной войны» угроза ядерного уничтожения заставила военных и политических лидеров очень серьезно отнестись к таким рискам. Легко

утверждать, что утечка данных может угрожать самому существованию страны. Оборонное сообщество требовало уровня компьютерной безопасности, намного превышающего то, что нужно бизнес сообществу.

Термин «многоуровневый» используется, потому что оборонное сообщество классифицировало людей и информацию на разные уровни доверия и чувствительности. Эти уровни представляют известные классификации безопасности: «конфиденциальные», «секретные» и «сверхсекретные». Прежде чем людям будет разрешено ознакомиться с «секретной» информацией, им должны быть предоставлены индивидуальные разрешения, основанные на индивидуальных расследованиях, чтобы установить их надежность. Люди, получившие «конфиденциальное» разрешение, имеют право видеть конфиденциальные документы, но им не доверяют смотреть на «секретную» или «сверхсекретную» информацию больше, чем кому-либо из широкой общественности. Эти уровни образуют простую иерархию, показанную на рисунке 1.1. Пунктирные стрелки на рисунке иллюстрируют направление, в котором правила позволяют передавать данные: от «более низких» уровней к «более высоким» уровням, а не наоборот.

Рисунок 1.1 – Иерархия уровней безопасности

Говоря об этих уровнях, обычно используются три разных термина:

- уровень допуска указывает уровень доверия, оказанного лицу, имеющего допуск, или компьютеру, который обрабатывает секретную информацию, или области, которая была физически защищена для хранения секретной информации. Уровень указывает на самый высокий уровень секретной информации, которая будет храниться или обрабатываться человеком, устройством или местоположением;
- уровень классификации указывает уровень чувствительности, связанный с некоторой информацией, например, в документе или файле компьютера. Предполагается, что уровень указывает на степень ущерба, который может последовать, если информация будет утеряна или раскрыта;

- уровень безопасности – это общий термин для уровня допуска или уровня классификации.

Оборонное сообщество было первым и крупнейшим заказчиком вычислительной техники, и компьютеры все еще были очень дорогими, когда они стали обычным оборудованием в оборонных организациях. Однако лишь немногие организации могут позволить себе использовать отдельные компьютеры для обработки информации на всех уровнях: им приходится разрабатывать процедуры совместного использования компьютера без утечки секретной информации неочищенным (или недостаточно очищенным) пользователям. Это не так просто, как может показаться. Даже когда пользователи «по очереди» запускали компьютер на разных уровнях безопасности (метод, называемый обработкой периодов), сотрудники службы безопасности должны были беспокоиться о том, оставалась ли «сверхсекретная» информация в памяти или на жестком диске операционной системы. Некоторые сайты приобрели компьютеры, чтобы посвятить их исключительно секретной работе, несмотря на стоимость, просто потому, что они не хотели рисковать утечкой информации.

Многопользовательские системы, как и ранние системы распределения времени, сделали такой обмен особенно сложным. В идеале, люди с «секретными» разрешениями должны иметь возможность работать в то же время, когда другие работают над «сверхсекретными» данными, и каждый должен иметь возможность обмениваться общими программами и несекретными файлами. В то время как типичные механизмы операционной системы, как правило, могут защитить различные пользовательские программы друг от друга, они не могут предотвратить «конфиденциального» или «секретного» пользователя от обмана «сверхсекретного» пользователя в выпуске сверхсекретной информации с помощью троянского коня.

Троянский конь - это программа, которая выполняет невидимую функцию, которую пользователь не выбрал бы для выполнения. Например, рассмотрим многопользовательскую систему, в которой пользователи хранят множество личных файлов и используют разрешения доступа системы для защиты этих файлов от посторонних глаз. Представьте себе, что автор локально разработанной программы обработки текстов имеет нездоровое любопытство о других в сообществе пользователей и хочет прочитать их защищенные файлы. Автор может установить функцию «троянского коня» в программе обработки текстов для извлечения защищенных файлов. Функция копирует личные файлы пользователя в собственный каталог автора всякий раз, когда пользователь запускает программу обработки текстов.

К сожалению, это не теоретическая угроза. Функция «макрос» в современных текстовых процессорах, таких как Microsoft Word, позволяет пользователям создавать произвольно сложные программные процедуры и прикреплять их к текстовым документам. Когда другой пользователь открывает документ, содержащий макрос, текстовый процессор выполняет процедуру, определенную макросом. Это было основой «макровирусов»,

таких как вирус «Я тебя люблю» конца 1990-х годов. Макрос может выполнять все функции, необходимые для программы троянского коня, включая копирование файлов.

Когда пользователь запускает программу обработки текстов, программа наследует права доступа этого пользователя к собственным файлам пользователя. Таким образом, троянский конь обходит права доступа, выполняя свою скрытую функцию, когда ничего не подозревающий пользователь запускает его. Это верно независимо от того, реализована ли функция в макросе или встроена в сам текстовый процессор. Вирусы и сетевые черви являются троянскими конями в том смысле, что их логика репликации выполняется в контексте зараженного пользователя. Иногда черви и вирусы могут включать дополнительный механизм троянского коня, который собирает секретные файлы от своих жертв. Если жертвой троянского коня является кто-то с доступом к сверхсекретной информации в системе с менее очищенными пользователями, то в обычной системе нет ничего, чтобы предотвратить утечку сверхсекретной информации. Многопользовательские системы явно нуждаются в специальном механизме защиты многоуровневых данных от утечки.

1.2Многопользовательские режимы работы

В Соединенных штатах Америки оборонное сообщество обычно описывает многопользовательскую систему как работающую в определенном режиме. В целях этого обсуждения существует три важных режима работы:

- специальный режим (dedicatedmode) - все пользователи в настоящее время в системе имеют разрешения на доступ к данным в системе. В специальном режиме сам компьютер не нуждается в каких-либо встроенных механизмах контроля доступа, только если заблокированные двери или другие физические механизмы препятствуют несанкционированному доступу к нему;
- системный высокий режим (systemhighmode) - все пользователи в настоящее время в системе имеют право доступа к любым данным в системе, но не все пользователи должны знать все данные. Если пользователям не нужно знать некоторые данные, система должна иметь механизмы для ограничения их доступа. Это требует типичных механизмов доступа к файлам типичных многопользовательских систем;
- многоуровневый режим (multilevelmode) - не все пользователи в настоящее время в системе очищены для всех данных, хранящихся в системе. Система должна иметь механизм контроля доступа, который применяет ограничения MLS. Также должны механизмы для применения ограничений доступа к многопользовательским файлам.

Фраза «необходимо знать» относится к обычно применяемому правилу в организациях, которые обрабатывают секретную информацию. В целом, разрешение безопасности не дает общего разрешения на просмотр всей информации, классифицированной на этом уровне или ниже. Допуск

действительно только первый шаг: людям разрешено смотреть только на секретную информацию, которую они должны знать, как часть работы, которую они делают.

Другими словами, если мы дадим одному пользователю секретный допуск для работы на криптографическом устройстве, то он должен знать секретную информацию, связанную с этим устройством. У него нет разрешения на изучение секретной информации о спутниках-шпионах или секретной криптографической информации, которая не относится к его устройству. Если он использует многопользовательскую систему, содержащую секретную информацию о своем проекте, других криптографических проектах и даже спутниках-шпионах, то система должна предотвратить просмотр данного пользователя информации, принадлежащей другим проектам и мероприятиям. С другой стороны, система должна быть в состоянии предоставить ему разрешение на просмотр других материалов, если он действительно нуждается в информации для выполнения своей работы.

Режим работы компьютера определяет, какие механизмы контроля доступа ему необходимы. Специальные системы могут не требовать каких-либо механизмов помимо физической безопасности. Компьютеры, работающие под управлением system high, должны иметь пользовательские ограничения доступа, подобные тем, которые обычно предоставляются в Unix и в «профессиональных» версиях Microsoft Windows. В многоуровневом режиме система должна предотвращать утечку данных с более высоких уровней безопасности пользователям с более низкими разрешениями: для этого требуется специальный механизм.

Как правило, механизм MLS работает следующим образом: пользователи, компьютеры и сети переносят читаемые компьютером метки, чтобы указать уровни безопасности. Данные могут передаваться «из того же уровня» к «тому же уровню» или от «более низкого уровня» к «более высокому уровню» (рисунок 1.2). Таким образом, сверхсекретные пользователи могут обмениваться данными друг с другом, а сверхсекретный пользователь может получать информацию от секретного пользователя. Это не позволяет данным из сверхсекретного (более высокого уровня) течь в файл или другое место, видимое для секретного пользователя (на более низком уровне). Если данные не подпадают под действие правил классификации, они относятся к «несекретному» уровню безопасности. На компьютере это будет включать в себя большинство прикладных программ и любые вычислительные ресурсы, общие для всех пользователей.

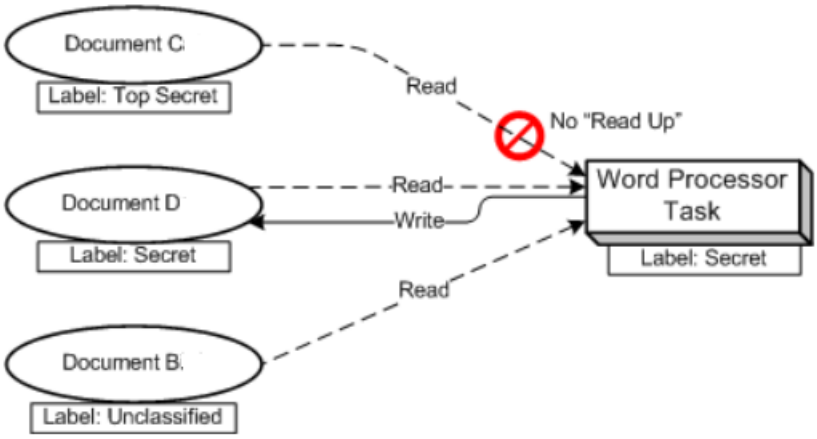

Рисунок 1.2 - Потоки данных, разрешенные механизмом MLS

Прямая реализация такой системы позволяет автору сверхсекретного отчета извлекать информацию, введенную пользователями, работающими в режиме секретности или конфиденциальности, и объединять ее с сверхсекретной информацией. Пользователь с секретным разрешением не может «читать», чтобы увидеть сверхсекретный результат, так как данные передаются только в одном направлении между секретным и сверхсекретным. Несекретные данные могут быть доступны всем пользователям.

Недостаточно просто запретить пользователям с более низкими разрешениями считывать данные с более высокими классификациями. Что делать, если пользователь с совершенно секретным разрешением хранит некоторые сверхсекретные данные в файле, читаемом секретным пользователем? Это вызывает ту же проблему, что и «чтение», так как оно делает сверхсекретные данные видимыми для секретного пользователя. Некоторые могут утверждать, что сверхсекретным пользователям следует доверять. На самом деле, некоторые утверждают, что они никогда этого не сделают, потому что это нарушение закона о шпионаже. К сожалению, этот аргумент не принимает во внимание риск «троянского коня».

Например, на рисунке 1.3 показано, что может произойти, если злоумышленник вставит макрос-функцию с возможностью троянского коня в текстовый файл.

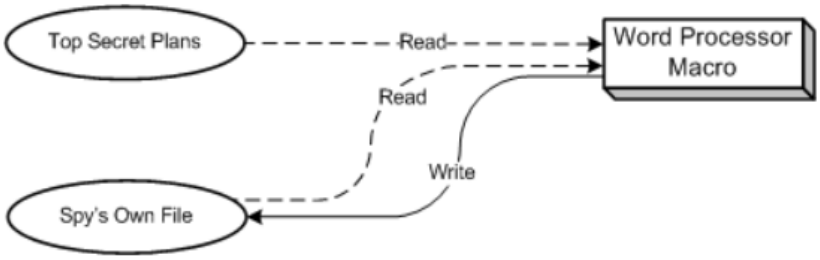

Рисунок 1.3 – Макрос-функция с троянским конем может привести к утечке данных на более низкий уровень безопасности

Злоумышленник сохранил функцию макроса в конфиденциальном файле и сообщил совершенно секретному пользователю о необходимости изучить файл. Когда пользователь открывает конфиденциальный файл, запускается макрос-функция, которая пытается скопировать файлы из сверхсекретного каталога пользователя в конфиденциальные файлы, принадлежащие злоумышленнику. Это называется «записыванием» данных с более высокого уровня безопасности на более низкий.

В системе с типичными механизмами контроля доступа макрос будет выполнен успешно, так как злоумышленник может легко настроить все разрешения, необходимые для записи данных в файлы другого пользователя. Очевидно, что система не может обеспечить MLS надежно, если троянские программы могут обойти защиту MLS. Нет никакого способа, которым пользователи могут избежать троянских программ с 100% надежностью, как это предполагает успех вирусов электронной почты. Эффективный механизм MLS должен блокировать попытки «записи», а также попытки «чтения».

1.3Модель Белл-ЛаПадула

Наиболее широко признанным подходом к MLS является модель безопасности Белл-ЛаПадула (Bell and La Padula, 1974). Модель эффективно отражает основы ограничений доступа, подразумеваемых обычными уровнями военной безопасности. Большинство механизмов MLS реализует модель Белл-ЛаПадула или ее близкий вариант [2]. Хотя модель точно определило возможность системы MLS, которая держит данные безопасными, это не привело к широко распространенной разработке успешных многоуровневых систем. На практике разработчики не смогли создать механизмы MLS, которые надежно работают с высокой уверенностью, а некоторые важные оборонные приложения требуют возможности «записать», что делает модель неуместной.

В модели Белл-ЛаПадула программы и процессы (называемые субъектами) пытаются передавать информацию через файлы, сообщения, устройства ввода-вывода или другие ресурсы в компьютерной системе (называемые объектами). Каждый субъект и объект имеет метку, содержащую его уровень безопасности, то есть уровень доступа субъекта или уровень классификации объекта. В простейшем случае уровни безопасности расположены в иерархии, как было ранее показано на рисунке 1.1. Более сложные случаи связаны с отсеками, как описано в одном из последующих разделов.

Модель Белл-ЛаПадула осуществляет ограничения доступа MLS, реализуя два простых правила (рисунок1.4):

- простое свойство безопасности (noreadup);
- *-свойство (no write down).

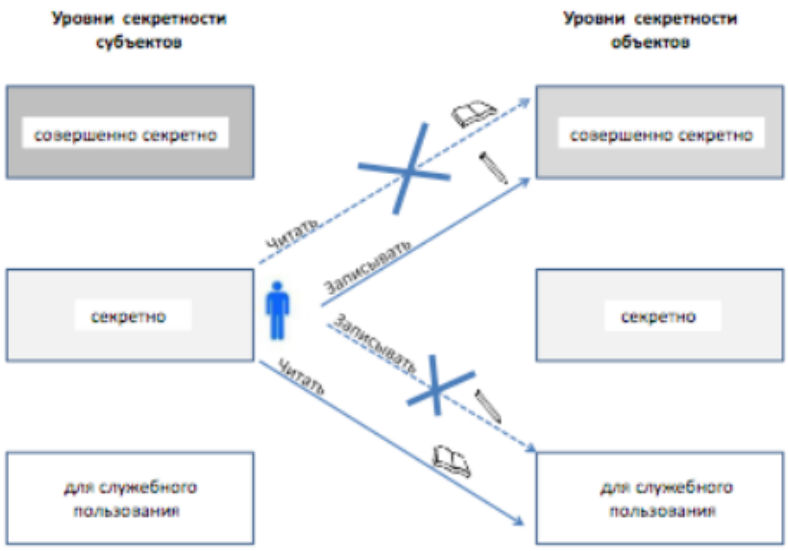

Рисунок 1.4 – Правила модели Белл-ЛаПадула

Когда субъект пытается прочитать или записать объект, система сравнивает метку безопасности субъекта с меткой объекта и применяет эти правила. В отличие от типичных ограничений доступа к многопользовательским вычислительным системам, эти ограничения являются обязательными: ни один пользователь системы не может их отключить или обойти. Типичные ограничения многопользовательского доступа являются дискреционными, то есть они могут быть включены или отключены системными администраторами и часто отдельными пользователями. Если пользователи или даже административные пользователи могут изменять правила доступа, троянский конь может изменять или даже отключать эти правила. Чтобы предотвратить утечку, просматривая пользователей и троянских коней, системы данной модели всегда применяют эти два свойства.

Простое свойство безопасности: субъект может считывать данные из объекта, если уровень безопасности субъекта совпадает с уровнем безопасности объекта или превышает его. Иногда это называется свойством «No read up».

*-свойство: субъект может выполнять запись в объект, если уровень безопасности субъекта равен или ниже уровня безопасности объекта. Иногда это называется свойством «No write down».

Простое свойство безопасности очевидно: оно не позволяет людям (или их процессам) читать данные, классификация которых превышает их разрешения безопасности. Пользователи не могут «читать вверх» относительно их уровней безопасности. Они могут «читать вниз», что означает, что они могут читать данные, классифицированные на или ниже того же уровня их безопасности.

*-свойство предотвращает пользователям с более высоким доступом от передачи данных высокой степени секретности для пользователей, которые не разделяют соответствующий доступ, либо случайно или намеренно. Пользовательские программы не могут «записывать» в файлы, которые имеют более низкий уровень безопасности, чем процесс, который они в настоящее время работают. Это предотвращает программы троянского коня от тайно утечки данных высокой секретности. Рисунок 5 иллюстрирует эти свойства: пунктирные стрелки показывают, что данные считываются в соответствии с простым свойством безопасности, а нижняя сплошная стрелка показывает попытку «записать», блокируемую *-свойством.

Система, применяющая модель Белл-ЛаПадула, блокирует макрос обработки текстов любым из двух способов в зависимости от уровня безопасности, на котором пользователь запускает программу обработки текстов. В одном случае, показанном на рисунке1.5, пользователь запускает программу Совершенно секретно. Это позволяет макросу считывать секретные файлы пользователя, но *-свойство предотвращает запись макроса в конфиденциальные файлы злоумышленника. Когда процесс пытается открыть конфиденциальный файл для записи, правила доступа MLS предотвращают его. В другом случае, совершенно секретный пользователь запускает программу на конфиденциальном уровне, так как файл является конфиденциальным. Функция макроса программы может считывать и изменять конфиденциальные файлы, включая файлы, настроенные злоумышленником, но простое свойство безопасности не позволяет макросу считывать секретные или сверхсекретные файлы.

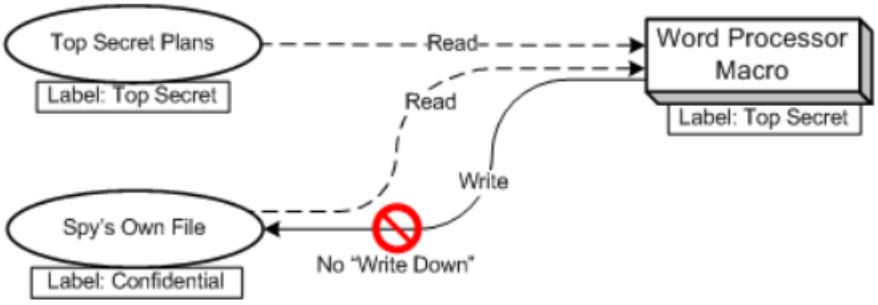

Рисунок1.5 – Если программа может считывать сверхсекретные данные, она не может записывать данные на более низкий уровень

Модель Белл-ЛаПадула (BLP) вызвала много волнений: прямая политика безопасности, которая была понятна интуитивному пониманию, но все же позволяла людям доказывать теоремы. Но Дж.Маклин показал, что правил модели самих по себе было недостаточно. Он представил SystemZ, определяемую как система модели BLP с добавленной функцией, что пользователь может попросить системного администратора временно рассекретить любой файл от высокого до низкого. Таким образом,

пользователи с низким доступом могут читать любой файл, не нарушая модель BLP.

Аргумент Белла заключался в том, что SystemZ обманывает, делая то, что модель не позволяет (изменение меток не является допустимой), и аргумент Дж.Маклина состоял в том, что он не сказал ему об этом. Вопрос решается путем введения свойства спокойствия. Свойство сильного спокойствия говорит, что метки безопасности никогда не меняются во время работы системы, в то время как свойство слабого спокойствия говорит, что метки никогда не меняются таким образом, чтобы нарушать определенную политику безопасности.

Мотивация для слабого свойства заключается в том, что в реальной системе мы часто хотим соблюдать принцип наименьших привилегий и начать процесс на неочищенном уровне, даже если владелец процесса был очищен до «совершенно секретно. Если она затем получает доступ к конфиденциальной электронной почте, ее сеанс автоматически обновляется до "конфиденциального"; и в целом, ее процесс обновляется каждый раз, когда он получает доступ к данным на более высоком уровне (это известно как принцип высокой отметки воды (highwatermark)). Поскольку субъекты обычно являются абстракцией подсистемы управления памятью и дескрипторов файлов, а не процессов, это означает, что состояние изменяется при изменении прав доступа, а не при фактическом перемещении данных.

Еще одна проблема с BLP моделью, да и со всеми системами обязательного контроля доступа, заключается в том, что разделение пользователей и процессов является относительно простым; трудная часть - это когда требуется некоторое контролируемое взаимодействие. Большинство реальных приложений нуждаются в каком-то «доверенном субъекте», который может нарушить политику безопасности; примером может служить доверенный текстовый процессор, который помогает аналитику разведки очистить сверхсекретный документ, когда она редактирует его до секретного. Модель BLP молчит о том, как система должна защищать такое приложение.

Наконец, стоит отметить, что даже с высокой степенью очистки воды модель BLP все еще не занимается созданием или уничтожением субъектов или объектов (что является одной из трудных проблем построения реальной системы MLS).

1.4Модель Биба

Включение в WindowsVistaмногоуровневой модели целостности возродил интерес к модели безопасности, разработанной в 1975 году Кеном Биба, которую в учебниках часто упоминается как «Белл-ЛаПадула вверх дном» или «инверсия модели Белл-ЛаПадула». В данной модели безопасного управления доступа под безопасностью понимается целостность.

Так же, как и модель Белл-ЛаПадулы, модель Биба (рисунок 1.6) оперирует субъектами и объектами, которые также разбиваются на

иерархически организованные уровни. Но вместо уровней секретности здесь вводятся уровни целостности. Чем выше уровень целостности объекта (например, документа), тем более он заслуживает доверия, тем выше вероятность, что он содержит точные данные, тем строже правила, допускающие субъекты к работе с этими данными. Чем выше уровень, к которому отнесен субъект, тем больше ему доверяют, в том числе по возможностям модификации информации, содержащейся в объектах.

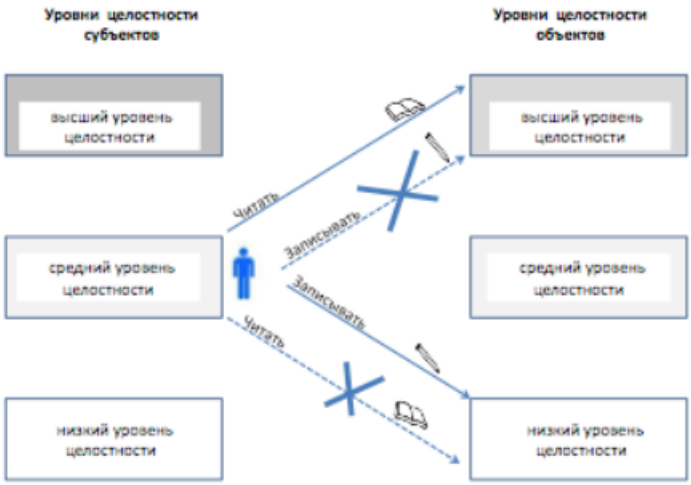

Рисунок 1.6 – Правила модели Биба

Субъекты выполняют над объектами операции «читать» и «записывать».

Модель Биба определяет два правила, при соблюдении которых система гарантированно будет находится в безопасном состоянии [2]:

- простая аксиома целостности (The Simple Integrity Axiom) - субъекту данного уровня целостности запрещено выполнять операцию «читать» по отношению к объектам более низкого уровня целостности (правило "no read down"). Субъект, читая данные из объекта, характеризуемого более низким уровнем целостности, рискует «испортить» данные своего уровня, сделать их менее достоверными, поэтому такие операции должны быть запрещены. Зато он может читать проверенную, более достоверную информацию с более высоких уровней;

- аксиома *-целостности (The *-Integrity Axiom) – субъекту данного уровня целостности запрещено выполнять операцию «записывать» по отношению к объектам более высокого уровня целостности (правило "no write up"). Субъект, доверие к которому ограничивается некоторым уровнем целостности, не должен иметь возможность записывать данные в объекты более высокого уровня, так как он сможет внести в них искажения, неточности и тем самым снизить безопасность системы. Поток данных субъекта, направленный «вниз», не может ухудшить степень целостности объектов, имеющих более низкий уровень целостности.

Как видим, правила модели Биба, направленные на обеспечение целостности данных, прямо противоположны правилам модели Белла-ЛаПадулы, гарантирующим конфиденциальность данных.

Как и следовало ожидать, у модели Бибы те же фундаментальные проблемы, что и у Белл-ЛаПадулы. Она не может приспособить реальную деятельность очень хорошо без многочисленных исключений. Например, реальная система обычно требует «доверенных» субъектов, которые могут переопределить модель безопасности, но модель Биба сама по себе не может обеспечить эффективные механизмы для их защиты и ограничения; и в целом она не так хорошо работает с современными программными средами. В конце концов, Vista отбросила ограничение No Read Down и не использовала свою модель целостности для защиты базовой системы от пользователей.

Помимо этих двух моделей разработаны также другие формальные модели безопасности (таблица 1.1).

Таблица 1.1 – Модели

Модель	Атрибуты	Политика	Примечание
Access Matrix	S (субъект),O (объект), accesses	DAC	строки:лист контроля столбцы: лист контроля доступа
BLP	S,O,a; no read up, no write down	DAC, MAC	
Biba	S,O,a; no read down, no write up	Authchanges (изменения в авторизации)	«кувырок» модели BLP
Clark Wilson	S,O,a; no read down, no write up	Auth changes, no mistakes, data consistency	хорошо сформированные транзакции,
Non Interference	Inputs (cmds), Outputs (views)	Auth changes, MAC	полезновССА, не решетка
Information Flow	Объекты, поток информации	Auth changes, MAC	полезновССА, не решетка

1.5Отсеки и системные привилегии

Несмотря на то, что иерархические уровни безопасности, такие как «сверхсекретно», знакомы большинству людей, они не являются единственными ограничениями, установленными для информации в оборонном сообществе. Организации применяют иерархические уровни безопасности (конфиденциальные, секретные или сверхсекретные) к своим данным в соответствии с ущербом, который может быть вызван утечкой этих данных. Некоторые организации добавляют другие маркировки к классифицированным материалам для дальнейшего ограничения их

распространения. Эти маркировки имеют много имен: отсеки, кодовые слова, предупреждения, категории, и так далее, и они служат многим целям. В некоторых случаях маркировка указывает, могут ли данные передаваться конкретным организациям, предприятиям или странам-союзникам. Во многих случаях эти метки дают создателю или владельцу данных больший контроль над распределением данных. Каждая маркировка указывает на другое ограничение, установленное для распределения определенного классифицированного элемента данных. Люди могут получать секретные данные только в том случае, если они соблюдают все ограничения, установленные на распространение данных.

Модель Белл-Ла-Падула относит все эти дополнительные обозначения как отсеки (compartments). Уровень безопасности может включать идентификаторы отсека в дополнение к иерархическому уровню безопасности. Если уровень безопасности конкретного файла включает один или несколько отсеков, то уровень безопасности пользователя также должен включать эти отсеки, иначе пользователю не будет разрешено читать файл.

Система с отсеками обычно получает большое количество определенных уровней обеспеченностью: одно для каждого законного сочетания из иерархического уровня безопасности с нулевым или больше отсеками. Взаимосвязи между этими уровнями образуют ориентированный граф, называемый решеткой. На рисунке 1.7 показана решетка для системы, содержащей секретную и сверхсекретную информацию с отсеками Ace и Bar.

Стрелки в решетке показывают, какие уровни безопасности могут считывать данные, помеченные другими уровнями безопасности. Если пользователь 1 имеет сверхсекретный доступ с доступом к обоим отсекам Ace и Bar, то у него есть разрешение на чтение любых данных в системе (при условии, что ее владелец также дал разрешение на «чтение» этих данных). Права доступа, связанные с другими метками безопасности, определяются стрелками на рисунке 1.7.

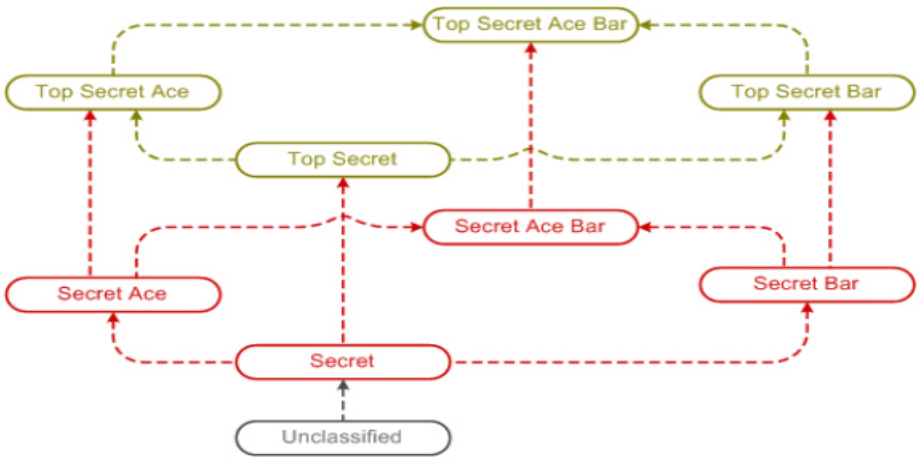

Рисунок 1.7 – Решетка с секретными и сверхсекретными данными с отсеками Ace иBar

Если пользователь 2выполняет программу с меткой Secret Ace, то программа может считывать данные с меткой Unclassified, Secret или Secret Ace. Программа не может читать данные с меткой Secret Bar или Secret Ace Bar, так как ее метка безопасности не содержит данного отделения. Рисунок 4 иллюстрирует это: нет никакого пути к секретному Ace, который прибывает из метки, содержащий Bar отсек. Точно так же программа не может читать сверхсекретные данные, потому что она работает на секретном уровне, и никакие сверхсекретные метки не приводят к секретным меткам.

MLSи системные привилегии.

Высокий уровень безопасности, такой как у пользователя 1, показывает, что конкретная организация готова доверять ему определенные типы секретной информации. Это не пустая проверка, которая предоставляет доступ к каждому ресурсу в компьютерной системе. Правила доступа MLSвсегда работают в сочетании с другими правилами доступа системы.

Системы, применяющие правила доступа MLS, всегда объединяют их со стандартными правами доступа, управляемыми пользователем. Если секретный или конфиденциальный пользователь блокирует доступ к файлу другим пользователям, то и совершенно секретный пользователь не может прочитать файл. Правило «необходимо знать» означает, что секретная информация должна предоставляться только тем лицам, которые действительно нуждаются в ней. Предполагается, что отдельные пользователи должны защищать секретную информацию от произвольного просмотра другими пользователями. Более высокие уровни доступа не дают права произвольно просматривать: доступ по-прежнему ограничен, нужно знать требования.

Пользователи, имеющие сверхсекретные и более высокие разрешения, не получают автоматически статус администратора или «супер пользователя» в многоуровневых компьютерных системах. В некотором смысле, совершенно секретный уровень безопасности фактически ограничивает то, что может сделать пользователь: программа, работающая в совершенно секретном режиме, не может установить несекретную прикладную программу, например. Многие административные задачи, такие как установка прикладных программ и других общих ресурсов, должны выполняться на неклассифицированном уровне. Если администратор устанавливает программы во время работы на сверхсекретном уровне, то программы будут устанавливаться с сверхсекретными метками. Пользователи с более низкими разрешениями не будут авторизованы для просмотра программ.

1.6 Исторические примеры систем MLS

После ряда исследовательских продуктов в конце 1970-х годов, в начале 1980-х годов начали внедряться продукты, которые применяли многоуровневые политики безопасности. К 1988 году ряд компаний начали внедрять версии MLS в своих операционных систем. Концепции MLS были

распространены на все виды продуктов. Ниже будут приведены и описаны основные системы MLSтого времени.

1.6.1SCOMP

Одним из наиболее важных продуктов был защищенный коммуникационный процессор (SCOMP), производный от Multics, выпущенный в 1983 году. Это была беспроцентная реализация того, что Министерство обороны США считало необходимым для обработки сообщений на нескольких уровнях классификации. Система имела официально подтвержденное аппаратное и программное обеспечение с минимальным ядром и четырьмя кольцами защиты (а не семь Multics), чтобы все было просто. Своя операционная система, STOP, использовала эти кольца для поддержания до тридцати двух отдельных отсеков, и позволяла получать односторонние информационные потоки между ними.

SCOMP был использован в таких приложениях, как военная почта охранников. Это специализированные брандмауэры, которые обычно позволяют почте переходить от низкого к высокому, но не наоборот. Преемник SCOMP, XTS-300, поддерживал C2G, Command and Control Guard. Это было использовано во время поэтапного развертывания сил данных системы (TPFDD), функция которого заключается в плане передвижения войск и соответствующей логистики. Военные планы разрабатываются в качестве TPFDD на высоком уровне классификации, а затем распределяются в соответствующее время в качестве командования на более низкие уровни для осуществления.

Самым значительным вкладом SCOMP было служить моделью для Оранжевой книги - критерия оценки надежных компьютерных систем США. Это был первый систематический набор стандартов для защищенных компьютерных систем, который был введен в 1985 году и окончательно отменен в декабре 2000 года. Оранжевая книга имела огромное влияние не только в США, но и среди союзных держав; такие страны, как Великобритания, Германия и Канада основывали на ней свои собственные Национальные стандарты, пока эти Национальные стандарты не были, наконец, включены в «Общие Критерии».

«Оранжевая книга» позволяет оценивать системы на нескольких уровнях, причем A1 является самым высоким и движется вниз через B3, B2, B1 и C2 к C1. SCOMP была первой системой с рейтингом A1. Система также широко документирована в открытой литературе. Будучи первой и довольно публичной, система установила стандарт для следующего поколения военных систем. Этот стандарт с тех пор можно было редко встретить; в действительности, XTS-300 было оценено только к B3 уровню.

1.6.2 BLACKER

Blacker был серией устройств шифрования, предназначенных для включения MLS технологии. Ранее устройства шифрования были созданы с отдельными процессорами для зашифрованного текста или Black, конца и открытого текста, или Red, конца. Различные возможные отказы могут быть предотвращены, если можно согласовать обработку Red и Black. Можно также упростить работу устройства и обеспечить более эффективную работу гибкости: устройство не ограничивается разделением двух логических сетей, но может обеспечивать выборочное шифрование и проверку целостности и взаимодействовать в полезных пути с маршрутизаторами. Но тогда требуется высокий уровень уверенности в том, что «Red» данные не будут просачиваться через «Black».

В 1989 году Blacker вошел в реализацию, и основным уроком, извлеченным из него, былочрезвычайная сложность размещения административного трафика в рамках модели уровней классификации. Уже в 1994 году это было единственное сообщение устройство безопасности с оценкой A1. Так что это тоже повлияло на последующие системы. Однако он не был широко использован, и его преемник (Motorola Network Encryption System), имел оценку B2.

1.6.3MLSUNIXи рабочие станции с дистанционным управлением

Версии MLS Unix начали появляться в конце 1980-х, например, от AT&T System V/MLS. Это добавило уровни безопасности и метки, первоначально используя некоторые биты в записи идентификатора группы и позже, используя это, чтобы указать на более сложную структуру, что позволило ввести свойства MLS с минимальными изменениями ядра системы. Каждая копия Solaris содержит механизмы MLS, для тех пользователей, кто достаточно хорошо осведомлен, чтобы использовать их.

Рабочие станции с дистанционным управлением (CMWs) является примером клиентов систем MLS. Они позволяют оператору-человеку одновременно просматривать и изменять данные на различных уровнях и обеспечивают надлежащее обновление меток, прикрепленных к информации. Первоначальное требование поступило от разведывательногосообщества, чьи аналитики могут иметь доступ к «сверхсекретным» данным, таким как расшифровки и отчеты агентов, и производить отчеты на «секретном» уровне для пользователей, таких как политические лидеры и офицеры на местах. Поскольку эти отчеты уязвимы для захвата, они не должны содержать никакой информации, которая поставила бы под угрозу источники и методы разведки.

CMWs позволяет аналитику просматривать «совершенно секретные» данные в одном окне, составлять отчет в другом и иметь механизмы для предотвращения случайного копирования первого в последнее (т. е. вырезать

и вставить работы из «секретного» в «совершенно секретный», но не наоборот).

1.6.4 NRLPUMP

Вскоре выяснилось, что простые почтовые безопасности и криптографические ящики были слишком ограничительными, так как помимо почты было разработано еще много сетевых сервисов. Традиционные механизмы MLS (такие как слепые записи и периодические сбои) неэффективны для служб реального времени.

Поэтому научно-исследовательская лаборатория ВМС США (NRL) разработала Pump - одностороннее устройство передачи данных (диод данных) для обеспечения безопасного одностороннего потока информации (рисунок 1.7).

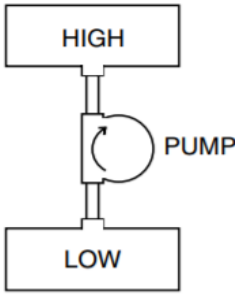

Рисунок 1.8 – NRLPump

Основная проблема заключается в том, что при отправке данных с низкого на высокий уровень, потребность в гарантированной надежности передачи означает, что сообщения подтверждения должны быть отправлены обратно с высокого на низкий. Pump ограничивает ширину полосы частот возможной отсталой утечки используя несколько механизмов как использование буферизации и рандомное время подтверждений. Привлекательность этого подхода заключается в том, что можно построить системы MLS, используя Pump для подключения отдельных систем на разных уровнях безопасности. Поскольку эти системы не обрабатывают данные более чем на одном уровне, они могут быть построены из дешевых коммерческих компонентов (COTS). Поскольку стоимость оборудования падает, это часто предпочтительный вариант.

1.6.5SYBARDSUITE

Большинство агентств информационной безопасности стран не смогли противостоять требованиям пользователей запускать стандартные приложения (например, MS Office), которые недоступны для многоуровневых безопасных платформ. Одним из ответов был «PurplePenelope», от Qinetiq в

Великобритании, теперь продается как Sybard Suite. Эта система помещает оболочку MLS вокруг рабочей станции Windows, реализуя высокую водяной знак версии BLP. Он отображает в фоновом режиме текущую безопасность уровня устройства и при необходимости обновляется, когда его более чувствительные ресурсычитаются. Это гарантирует правильность маркировки полученного продукта.

Вместо того чтобы препятствовать пользователям понижать рейтинг, как классическую систему BLP возможно, он позволяет им назначать любую метку безопасности, которая им нравится на их выходе. Однако если это связано с понижением, пользователь должен подтвердить выпуск данных с использованием интерфейса доверенного пути, что гарантирует отсутствие троянских коней или вирусов, чтоможет высвободить что-то совершенно незамеченное. Существует также аудит,чтобы обеспечить запись всех понижений, чтобы ошибки и атаки (независимо от того, пользователями или вредоносными программами) можно проследить после факта.

1.7MLS системы будущего

1.7.1 VISTA

Многоуровневая целостность приходит к массовому рынку в перспективеVista. Перспектива существенно использует модель Биба. Все процессы, и всезащищаемые объекты (включая каталоги, файлы и разделы реестра) имеютметку уровня целостности. По умолчанию файловые объекты помечаются как «средние», aInternetExplorer (и все загруженные с его помощью) помечаются как «низкий».Поэтому действия пользователя необходимы для обновления загруженного содержимого, прежде чем оно сможет изменять существующие файлы. Это не может быть панацеей: это может стать таким обычным требованиемиз всего установленного программного обеспечения, что пользователи будут обучены кротко обновления вирусов тоже по запросу. И следует иметь в виду, что большая часть шпионских программ, заражающих средний домашний компьютер, была установлена там преднамеренно (хотя и небрежно и с неполного знания последствий) после посещения какого-либо коммерческого веб-сайта. Это перекрытие между желаемым и нежелательным программным обеспечением устанавливает ограничение на то, сколько может быть достигнуто против загруженных вредоносных программ.

Кроме того, можно реализовать политику BLP с помощью Vista, или «No Read Up» и «No Execute Up» политики. Они не установлены по умолчанию; причина, по-видимому, в том, что Microsoft была главным образом обеспокоена установкой вредоносных программ у себя в системе, а затем скрывается. Держать браузер "низкий" делает установку сложнее, и позволяя всем процессам (даже низким), чтобы проверить остальную часть системы делает скрытие сложнее. Но это означает, что вредоносные

программы, работающие на низком уровне, могут украсть все ваши данные; поэтому некоторые пользователи могут установить «NoRead Up» для конфиденциальных каталогов. Без сомнения, эторазбейте несколько приложений, поэтому осторожный пользователь может позаботиться о том, чтобы иметь отдельные учетные записи для просмотра веб-страниц, электронной почты и конфиденциальных проектов. Это все обсуждается Рутковской; она также описывает некоторые интересные возможности атаки на основе виртуализации. Еще одна проблема заключается в том, что Vista, в защищенном режиме, по-прежнему пишет в высокой целостности части реестра, даже если Microsoft говорит, что этого не должно делаться.

Попутно, также стоит упомянуть управление правами, будь то классический вид DRM или более поздняя разновидность IRM, как случай обязательного контроля доступа. Vista, например, пытается гарантировать, что видео контент высокой четкости никогда не будет доступен ненадежному процессу.

1.7.2 LINUX

Случай SELinux и Red Hat несколько похож на Vista в том, что непосредственной целью новых механизмов обязательного контроля доступа является также ограничение последствий компромисса. SELinux основан на архитектуре безопасности колбы (Flask), которая отделяет политику от механизма принуждения; контекст безопасности содержит все атрибуты безопасности, связанные с субъектом или объектом в Flask, где один из этих атрибутов включает атрибут Type Enforcement. Идентификатор безопасности - это дескриптор контекста безопасности, сопоставленный сервером безопасности. Он имеет сервер безопасности, где принимаются решения политики, это находится в ядре, так как Linux имеет монолитное ядро, и разработчики не хотели требовать вызова пространства пользователя ядра для решений по безопасности (особенно, поскольку некоторые из них происходят на критических путях, где ядро удерживает блокировки). Сервер, который предоставляет общий API безопасности остальной части ядра, с моделью безопасности, скрытой за этим API. Сервер внутренне реализует RBAC, TE и MLS (или, если быть точным, механизм общих ограничений, который может выразить MLS или любую другую модель, которую вам нравится). SELinux включен в ряд дистрибутивов Linux, и использование Red Hat является типичным. Там его функция заключается в разделении различных услуг. Таким образом, злоумышленник, который захватывает ваш веб-сервер, не получает ваш DNS-сервер.

1.8Проблема обеспечения MLS

Несмотря на сильную поддержку со стороны военного сообщества и значительные усилия со стороны поставщиков вычислительной техники и

исследователей компьютерной безопасности, механизмы MLS не смогли обеспечить безопасность и функциональность, необходимые оборонному сообществу. Во-первых, исследователи безопасности и разработчики систем MLS обнаружили, что полностью предотвратить поток информации между различными уровнями безопасности в системе MLS чрезвычайно сложно и, возможно, невозможно. Вторая проблема заключалась в вирусной угрозе: когда мы осуществляем информационный поток MLS, мы ничего не делаем, чтобы предотвратить распространение вируса, введенного на более низком уровне доступа, на более высокие уровни. Наконец, сообщество конечных пользователей выявило ряд случаев, когда модель информационного потока Белл-Ла-Падула не полностью удовлетворяла их оперативные потребности и потребности в области безопасности.

Самовоспроизводящихся программ, как компьютерные вирусы стало незначительным явлением среди первых пользователей домашнего компьютера в конце 1970-х годов. В то время вирусы не были достаточным феноменом, чтобызаинтересовать MLS исследователей и разработчиков, системы MLS заинтересовали новаторского вирусного исследователя Фред Коэна (1990, 1994). В 1984 году он продемонстрировал, что вирус, введенный на неклассифицированном (несекретном) уровне системы, в которой реализована модель Белл-Ла-Падула, может быстро распространяться по всем уровням безопасности системы. Это конкретное заражение не отразило ошибку в реализации MLS. Вместо этого это указало на недостаток в модели Белл-Ла-Падула, которая стремится обеспечить потоки информации от низкого до высокого, предотвращая при этом потоки от высокого до низкого. Вирусы представляют угрозу безопасности, которая использует поток информации от низкого до высокого, поэтому защита MLS на основе модели Белл-Ла-Падула не дает никакой защиты от него.

Вирусы представляли собой один случай, когда модель Белл-Ла-Падула не удовлетворяла оперативным потребностям и потребностям безопасности конечных пользователей. Дополнительные случаи появились, когда конечные пользователи получили опыт работы с системами MLS. Одна из проблем заключалась в том, что системы, как правило, собирали много «сверх классифицированной» информации. Всякий раз, когда пользователь создает документ на высоком уровне безопасности, документ должен будет сохранить этот уровень безопасности, даже если пользователь удалил всю конфиденциальную информацию, чтобы создать менее классифицированный или даже неклассифицированный (несекретный) документ. В сущности, конечным пользователям часто нужен был механизм «понижения» информации, чтобы ее метка отражала ее пониженную чувствительность.

Проблема понижения стала особенно важной, поскольку конечные пользователи стремились разработать системы «датчик к стрелку». Эти системы использовали бы сверхсекретные данные разведки для получения тактических команд для отправки в боевые части которого рации получили информацию на секретный уровень или ниже. На практике системы будут

решать проблему понижения уровня путем установки привилегированных программ, которые обходят механизм MLS для понижения уровня информации. Хотя это послужило удобным исправлением для решения проблемы, оно также показало, что практические системы не полностью полагаются на механизмы Белл-Ла-Падула, которые стоили так много для сборки и проверки. Это еще больше подорвало интерес оборонного сообщества к MLS на основе продуктов Белл-Ла-Падула.

Члены оборонного сообщества определили потребность в системах, способных работать с MLS в 1960-х годах, и несколько поставщиков реализовали основные функции (Weissman 1969, Hoffman 1973, Karger и Schell 1974). Однако правительственные исследования проблемы MLS подчеркнули опасность использования больших, непрозрачных операционных систем для защиты действительно ценных секретов (Ware 1970, Anderson 1972). Операционные системы уже были печально известны своей ненадежностью, и эти сообщения высветили угрозу ошибки в программном обеспечении, позволяющей утечки высокочувствительной информации. Рекомендованное решение заключалось в том, чтобы обеспечить высокую степень уверенности путем проведения обширного анализа, обзора и тестирования.

Высокая гарантия, несомненно, увеличит затраты поставщиков на разработку и приведет к более высоким затратам на продукцию. Это не сдерживало оборонное сообщество США, которое предвидело долгосрочную экономию средств. Каргер и Шелл (1974) повторили утверждение о том, что возможности MLS могут спасти только ВВС США $100,000,000 в год, основываясь на вычислительных расходах в то время.

Каждое устройство MLS ставит фундаментальный вопрос: действительно ли оно обеспечивает MLS, или оно каким-то образом пропускает информацию? Первая задача MLS - разработать способ ответа на этот вопрос. Мы можем разложить проблему на два вопроса:

- Что на самом деле означает обеспечение MLS?
- Как мы можем оценить систему, чтобы убедиться, что она осуществляет MLS?

На первый вопрос ответила разработка моделей безопасности, таких как модель Белл-Ла-Падула, обобщенная ранее. Истинная модель безопасности обеспечивает формальное математическое представление ограничений потока информации MLS. Формальная модель делает проблему право применения ясной для не-программистов. Это также ясно дает понять эксплуатационное требование к программистам, которые реализовали механизмы MLS.

Чтобы решить вопрос оценки, разработчикам нужен был способ доказать, что элементы управления MLS системы действительно работают правильно. К концу 1960-х годов, это стало действительно серьезной проблемой. Программные системы стали слишком большими, чтобы кто-либо мог их проверить: Брукс (1975) сообщил, что IBM имеет более тысячи человек, работающих над своей новаторской системой OS/360. В книге

«Мифический человек-месяц» Брукс описал трудности построения масштабной программной системы. Размер проекта не был единственной проблемой в создании надежного и безопасного программного обеспечения: небольшие команды, такие как команда, ответственная за механизмы безопасности Multics, не могли обнаружить и закрыть каждую Уязвимость (Karger and Schell, 1974).

Сообщество безопасности разработало два набора стратегий для оценки систем MLS:

- стратегии для разработки надежной системы MLS;
- стратегии для доказательства правильной работы системы MLS.

В стратегиях разработки была подчеркнута особая структура, обеспечивающая единообразное применение правил доступа к данным, называемая эталонным монитором. Стратегии проектирования также требовали, чтобы проектировщики явно идентифицировали все компоненты системы, которые играли роль в обеспечении MLS; эти компоненты были определены как часть доверенной вычислительной базы, которая включала все компоненты, которые требовали высокой уверенности.

Стратегии доказывания правильности в значительной степени опирались на формальные проектные спецификации и методы анализа этих проектов. Некоторые из этих стратегий были реакцией на продолжающиеся проблемы контроля качества в индустрии программного обеспечения, но другие были разработаны как попытка обнаружить скрытые каналы, в основном нерешенную слабость в системах MLS.

1.8.1 Стратегии разработки системы MLS

В начале 1970-х годов ВВС США заказали исследование для разработки возможных стратегий построения и проверки систем MLS. Исследование объединило важные выводы исследователей безопасности в то время в отчет, названный Андерсонским отчетом (Anderson, 1972), который сильно повлиял на последующую поддержку правительством США систем MLS. В одном из последующих исследований (Nibaldi 1979) были определены наиболее перспективные стратегии разработки надежных систем и предложен набор критериев оценки таких систем.

Эти предложения привели к опубликованию критериев для разработки и оценки систем MLS, называемых критериями оценки надежных компьютерных систем (TCSEC) или «оранжевой книгой» (Министерство обороны, 1985). Правительство США установило процесс, с помощью которого поставщики компьютерных систем могут представить свои продукты для оценки безопасности. Правительственная организация, Национальный центр компьютерной безопасности (NCSC), оценила продукты против TCSEC и оценила продукты согласно их возможностям и степени доверия. Для того, чтобы продукт достиг наивысшей оценки надежности, NCSC необходимо было проверить правильность проекта продукта.

Чтобы сделать проверку проекта возможной, отчет Андерсона рекомендовал (и TCSEC требовал), чтобы системы MLS обеспечили безопасность через «механизм проверки ссылки», который сегодня мы называем контрольным монитором. Эталонный монитор является центральной точкой, которая обеспечивает все разрешения на доступ. В частности, эталонный монитор должен иметь три функции:

- должен быть защищен от несанкционированного доступа - злоумышленники или другие пользователи системы не должны намеренно или случайно отключать его или иным образом вмешиваться в его работу;
- не должен быть проходимым (nonbypassable) - все доступы к системным ресурсам должны осуществляться на контрольный монитор. Получить доступ к системным ресурсам можно только с помощью механизмов, использующих эталонный (ссылочный) монитор для принятия решений по управлению доступом;
- должно поддаваться проверки - должен быть способ убедить сторонних оценщиков (таких как NCSC), что система всегда будет обеспечивать MLS правильно. Эталонный монитор должен быть небольшим и достаточно простым по своей конструкции и осуществлению, с тем чтобы сделать проверку практичной.

Разработчики операционных систем к тому времени уже распознали концепцию ядра операционной системы: часть системы, которая делает неограниченный доступ к ресурсам компьютера, чтобы другие компоненты не нуждались в неограниченном доступе. Многие разработчики считали, что хорошее ядро должно быть маленьким по той же причине, что и эталонный монитор: легче построить уверенность в небольшом программном компоненте, чем в большом. Это привело к концепции ядра безопасности: ядра операционной системы, включающего эталонный монитор. Многослойность на ядре безопасности будет поддерживать процессы и утилиты для обслуживания пользователей системы и администраторов. Некоторые программы, не являющиеся ядрами, требуют привилегированного доступа к системным ресурсам, но ни одно из них не обходит ядро безопасности. Комбинация компьютерного оборудования, ядра безопасности и его привилегированных компонентов составляет доверенную вычислительную базу (ДВБ) - системные компоненты, ответственные за обеспечение соблюдения ограничений MLS. ДВБ был в центре усилий по обеспечению: если бы он работал правильно, то система правильно применяла бы ограничения MLS.

1.8.2 Проверка правильности системы MLS

Исследования, завершившиеся в рамках TCSEC, выявили три основных элемента для обеспечения правильной работы системы MLS:

- политика безопасности - явное заявление о том, что должна делать система. Это основывалось на модели обеспечения безопасности, которая описывала, как система применяла MLS;
- механизмы безопасности – функции ссылочного монитора и доверительная вычислительная база, которая принудительно применяет политику;
- обеспечения - свидетельство того, что механизмы фактически осуществляют политику.

Компьютерная индустрия всегда полагалась главным образом на системном тестировании для обеспечения качества. Однако отчет Андерсона признал недостатки тестирования, повторив замечание Дейкстры о том, что тесты могут доказать только наличие ошибок, а не их отсутствие. Для повышения надежности в отчете содержатся конкретные рекомендации по проектированию, построению и тестированию систем MLS. Эти рекомендации стали требованиями в TCSEC, особенно для продуктов, предназначенных для наиболее важных применений:

- проектирование сверху вниз - дизайн продукта должен иметь спецификацию дизайна верхнего уровня и детальную спецификацию проектирования;
- формальная спецификация политики;
- формальная спецификация верхнего уровня - должна быть формальная спецификация внешнего видимого поведения продукта;
- доказательство правильности проектирования - должно быть математическое доказательство, показывающее, что проект верхнего уровня соответствует политике безопасности;
- спецификация на соответствие кода - должен быть способ показать, что механизмы, появляющиеся в формальной спецификации верхнего уровня, реализованы в исходном коде.

Этими действиями не заменили стандартные методы разработки продуктов. Вместо этого эти дополнительные задачи были объединены с общепринятыми «методами наиболее успешной практики», используемыми при разработке обычных компьютерных систем. Эти практики, как правило, следуют процессу «водопада» (Boehm, 1981; Министерство обороны, 1985b): сначала разработчики разрабатывают спецификацию требований, из которой они разрабатывают дизайн сверху вниз, затем они реализуют продукт, и, наконец, они тестируют продукт на соответствие требованиям. В идеализированном процессе разработки продукта MLS спецификация требований фокусируется на тестируемых функциях и измеримых возможностях производительности, в то время как модель политики фиксирует требования безопасности, которые не могут быть протестированы

напрямую. Рисунок 1.9 показывает, как эти элементы работали вместе для проверки правильной работы продукта MLS.

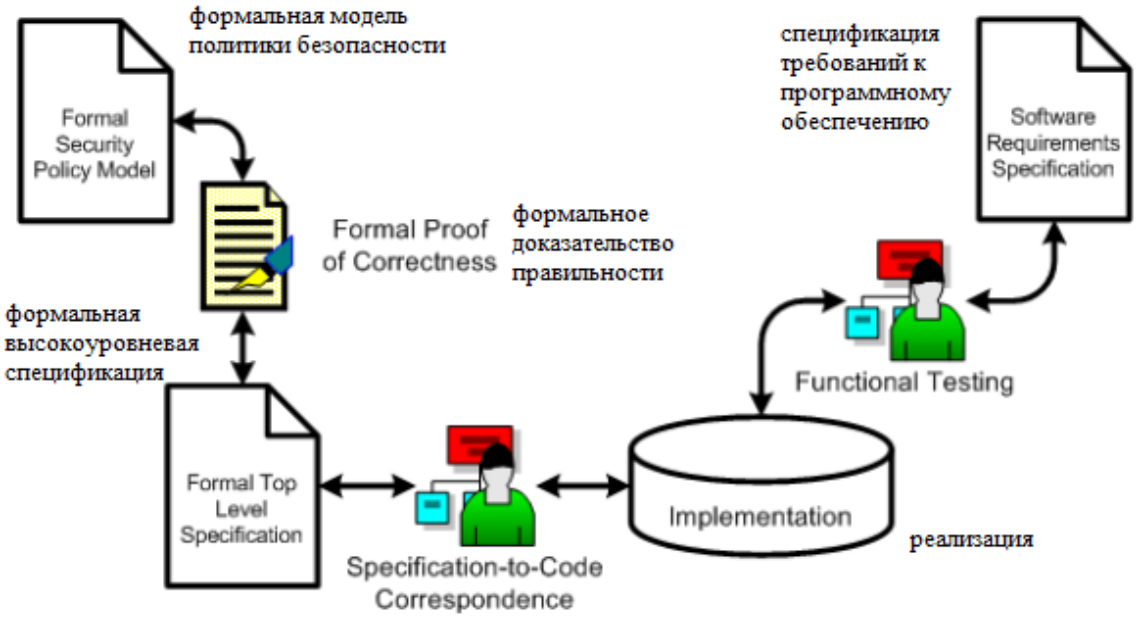

Рисунок 1.9 – Взаимосвязь политики, спецификаций, реализации и обеспечений

Разработка продукта всегда была дорогой. Многие организации по разработке, особенно небольшие, пытаются сэкономить время и деньги, пропуская этапы планирования и проектирования процесса водопада. TCSEC не требовал процесс водопада, но его требования для очень уверенных систем наложили значительные затраты на организации разработки. Оба исследования Nibaldi и TCSEC признали, что не все разработчики могут позволить себе добиться высочайшего уровня надежности. Вместо этого в процессе оценки был определен ряд уровней гарантий, которых может достичь тот или иной продукт. Продукты, предназначенные для менее важных видов деятельности, могут расходовать меньше средств на процесс их разработки и обеспечивать более низкий уровень гарантий. Вместе с тем ожидается, что продукция, предназначенная для наиболее важных областей применения, будет отвечать самым высоким стандартам практического обеспечения.

Скрытые каналы.

Вскоре после появления отчета Андерсона, Лэмпсон (1973) опубликовал примечание, которое исследовало общую проблему хранения информации в одной программе в секрете от другой, проблема в применения MLS. Лэмпсон отметил, что компьютерные системы содержат множество каналов, по которым два процесса могут обмениваться данными. Помимо явных каналов, таких как файловая система или службы межпроцессного взаимодействия, существуют скрытые каналы, которые также могут передавать данные между

процессами. Эти каналы обычно используют ресурсы операционной системы, совместно используемые всеми процессами. Например, когда один процесс может получить исключительный контроль над файлом, он не позволяет другим процессам получить доступ к файлу, или когда один процесс использует все свободное пространство на жестком диске, другие процессы будут «видеть» это действие.

Поскольку системы MLS не могли достичь своей основной цели (защитить секреты) при наличии скрытых каналов, эксперты по безопасности разработали методы обнаружения таких каналов. TCSEC требовало анализа скрытых каналов всех систем MLS, кроме самых низких уровней обеспечения.

В общем, есть две категории скрытых каналов: каналы с памятью (хранения) и каналы синхронизации. Канал хранения передает данные из «высокого» процесса в «низкий», записывая данные в место хранения, видимое «низкому». Например, секретный процесс может видеть, сколько памяти осталось после того, как сверхсекретный процесс выделяет некоторую память, сверхсекретный процесс может отправить числовое сообщение, выделив или освободив объем памяти, равный числовому значению сообщения. Скрытый канал состоит из установки содержимого места хранения (размера свободной памяти) на значение с помощью «высокого» процесса, которое читается «низким».

Канал синхронизации- это канал, в котором «высокий» процесс взаимодействует с «низким» путем изменения времени некоторого обнаруживаемого события. Например, совершенно секретный процесс может указать жесткому диску посетить определенные блоки дисков. Когда секретный процесс переходит к чтению данных с самого жесткого диска, активность диска сверхсекретным процессом вызовет различные задержки в секретной программе, когда она попытается использовать сам жесткий диск. Сверхсекретная программа может систематически накладывать задержки на дисковые операции секретной программы и, таким образом, передавать информацию через шаблон этих задержек. Врэй (1991) описывает скрытый канал на основе скорости доступа к жесткому диску, а также использует пример, чтобы показать, насколько неоднозначными могут быть две скрытые категории каналов.

Основная стратегия поиска каналов преобразования заключается в проверке всех общих ресурсов в системе, принятии решения о том, может ли какой-либо из них дать эффективный скрытый канал, и измерении пропускной способности любых скрытых каналов. В то время как случайная проверка квалифицированным аналитиком часто может выявить скрытые каналы, нет никакой гарантии, что случайная проверка найдет все такие каналы. Систематические методы помогают повысить уверенность в том, что поиск был всесторонним. Ранний метод, матрица общих ресурсов (Kemmerer, 1983; 2002), может анализировать систему из формальной или неформальной спецификации. В то время как метод может обнаружить скрытые каналы хранения, он не может обнаружить скрытые каналы синхронизации.

Альтернативный подход-невмешательство-требует формальной политики и спецификаций проекта (Haigh and Young, 1987). Этот метод находит как каналы синхронизации, так и каналы хранения, доказывая теоремы, чтобы показать, что процессы в системе, как описано в спецификации проектирования, не могут выполнять обнаруживаемые («вмешивающиеся») действия, которые видны другим процессам в нарушение ограничений MLS.

Чтобы эффективно обнаруживать скрытые каналы, спецификация проектирования должна точно моделировать весь общий доступ к ресурсам, видимый для пользовательских процессов в системе. Как правило, спецификация фокусирует свое внимание на системных функциях, доступных пользовательским процессам: системные вызовы для управления файлами, выделения памяти, взаимодействия с другими процессами и так далее. Программа развития системы блокировки (Сайдджари, Бекман и Лиман, 1989; Сайдджари, 2002), например, включала разработку официальной спецификации конструкции для поддержки анализа скрытых каналов. В спецификации проектирования блокировок были определены все системные вызовы, описаны все входы и выходы, создаваемые этими вызовами, включая результаты ошибок, и представлены внутренние механизмы, необходимые для поддержки этих возможностей. Команда блокировки использовала форму невмешательства для разработки доказательств того, что система правильно применяла MLS (Fine, 1994).

Как с любым методом обнаружения дефекта, нет никакогоспособа подтвердить, что все они были найдены. Методы, которые анализируют формальную спецификацию, обнаружат все недостатки в этой спецификации, но нет никакого способа окончательно доказать, что фактическая система прекрасно реализует спецификацию. Методы, основанные на менее формальных описаниях проекта, также ограничены качеством этих описаний: если описание пропускает функцию, нет способа узнать, открывает ли эта функция скрытый канал. В какой-то момент должен быть компромисс между усилиями, потраченными на поиск скрытых каналов, и усилиями, потраченными на поиск других недостатков системы.

На практике разработчики системы обнаружили, что практически невозможно устранить все скрытые каналы. Хотя критерии оценки побуждают разработчиков устранять как можно больше скрытых каналов, критерии также признают, что практические системы, вероятно, будут включать некоторые каналы. Вместо того, чтобы устранять каналы, разработчики должны идентифицировать их, измерять их возможную пропускную способность и предоставлять стратегии для снижения их потенциального ущерба. Хотя не все эксперты по безопасности согласны с тем, что скрытые каналы неизбежны (Proctor and Neumann, 1992), типичные продукты MLS содержат скрытые каналы. Таким образом, даже одобренные продукты MLS содержат известные недостатки.

1.9Оценка, сертификация и аккредитация

Существует три главных элемента в процессе утверждения, что система будет работать правильно, в оборонном сообществе США:

- оценка - проверяет набор свойств безопасности в конкретном продукте или устройстве. Оценка продукта не строго необходима;
- сертификация - проверка соответствия конкретной установки и конфигурации системы требованиям для данного сайта. Хотя это не является абсолютно необходимым, системы редко утверждаются для эксплуатации без, по крайней мере, попытки сертификации;
- аккредитация - официальное утверждение операции старшим военным командиром. Утверждение почти всегда основано на результатах сертификации.

В военных условиях высокопоставленный офицер, как правило, адмирал или генерал, должен официально предоставить разрешение (аккредитацию), прежде чем критическая система начнет функционировать. Аккредитация показывает, что, по мнению сотрудника, система безопасна в эксплуатации или, по крайней мере, риски системы перевешиваются ее преимуществами. Решение основано на результатах сертификации системы: процесс, в котором технические эксперты анализируют и тестируют систему, чтобы убедиться, что она отвечает требованиям безопасности. Процесс сертификации и аккредитации должен соответствовать определенным стандартам (Министерство обороны, 1997). В редких, чрезвычайных условиях сотрудник может аккредитовать систему, даже если есть проблемы с сертификацией.

Сертификация может быть очень дорогостоящей, особенно для систем MLS. Тесты и анализы должны показать, что система не потерпит неудачу таким образом, чтобы произошла утечка секретной информации или вмешательство в миссию организации. Тесты также должны показать, что все механизмы и процедуры безопасности работают в соответствии с требованиями. Сертификация системы специально часто подразумевает проектирование ревизий и проверок кода. Эта работа требует больших усилий и специальных навыков, что приводит к очень высоким затратам.

Процесс оценки продукта, объявленный TCSEC, был предназначен для обеспечения готового вычислительного оборудования, которое надежно обеспечивало ограничения MLS. Хотя организации могут внедрять, сертифицировать и аккредитовывать пользовательские системы, применяющие MLS, затраты на сертификацию трудно предсказать и могут превысить бюджет проекта. Если бы разработчики систем могли использовать готовые продукты MLS, их затраты на сертификацию и проектные риски были бы намного ниже. Сертификаторы могут полагаться на средства безопасности, проверенные во время оценки, вместо того, чтобы проверять реализацию продукта самостоятельно.

Оценки продуктов оценивают два основных аспекта: функциональность и надежность. Успешная оценка показывает, что продукт содержит

соответствующие функциональные характеристики и соответствует указанному уровню гарантии. TCSEC определил диапазон уровней оценки с учетом повышения уровня соответствия функциональным требованиям и требованиям гарантии. Каждый более высокий уровень оценки либо включал требования следующего более низкого уровня, либо заменял конкретные требования более жесткими требованиями. Буквенно-цифровые коды обозначают каждый уровень, причем D-самый низкий, а A1-самый высокий:

D - самый низкий уровень, присвоенный оцениваемым продуктам, не достигающим более высокого уровня; этот уровень использовался редко.

C1 - однопользовательская система; этот уровень в конечном итоге был отброшен.

C2 - многопользовательская система, как Unix Time sharing system.

C3 - усовершенствованная многопользовательская система; этот уровень в конечном итоге был отброшен.

B1 - самый низкий уровень оценки для системы с поддержкой MLS.

B2 - система MLS которая включает основные архитектурные требования обеспечения и анализ скрытого канала.

B3 - система MLS, которая включает в себя более значительные требования обеспечения, включая официальные спецификации.

A1 - система MLS, конструкция которой была доказана математически правильной.

Хотя в рамках TCSEC определен целый ряд уровней оценки, правительство хотело бы поощрять поставщиков к разработке систем, отвечающих самым высоким требованиям. Фактически, одним из первых оцененных продуктов был SCOMP, система A1, построенная Honeywell (Fraim, 1983). Очень немногие другие поставщики проводили оценку A1. Высокая гарантия привела к высоким затратам на разработку продукта; по оценкам одного проекта, высокие задачи обеспечения добавили 26% к трудозатратам на разработку (Smith, 2001). В быстро развивающемся мире разработки компьютерных продуктов эти дополнительные усилия могут привести к задержкам, которые делают разницу между успехом или неудачей продукта.

До настоящего времени никакие коммерческие производители компьютеров не предлагали подлинный «стандартный» продукт MLS. Несколько поставщиков внедрили операционные системы MLS, но ни один из них не был стандартным предложением продуктов. Вся продукция MLS была дорогостоящей, системы специального назначения продавались почти исключительно военным и государственным заказчикам. Почти все продукты MLS были оценены к уровню B1, соответствуя минимальным стандартам обеспечения. Таким образом, программа TCSEC потерпела неудачу на двух уровнях: она не смогла убедить поставщиков включить функции MLS в свои стандартные продукты, и она не смогла убедить каких-либо поставщиков производить продукты, которые отвечают требованиям A1 для высокой гарантии.

Обзор оценок продуктов безопасности, завершенный к концу 1999 года (Smith, 2000), показал, что только часть продуктов безопасности когда-либо проводила оценку. В большинстве продуктов проводились оценки «среднего уровня надежности», которых может быть достаточно для минимальной (B1) реализации MLS.

Оценки TCSEC были прекращены в 2000 году. Несколько современных продуктов MLS оцениваются по общим критериям (Common Criteria Project Sponsoring Organizations, 1999), критериям оценки, разработанным для решения более широкого спектра продуктов безопасности.

Наиболее заметным сбоем технологии MLS является ее отсутствие на типичных рабочих столах. Поскольку операционные системы Microsoft Windows стали доминировать на рабочем столе в 1990-х годах, Microsoft не сделала значительного шага для внедрения технологии MLS. Версии Windows заслужили оценку TCSEC C2 и более строгую оценку EAL-4 по общим критериям, но она никогда не включала MLS. Ближайший Microsoft пришел к предложению технологии MLS был его «палладий» усилия, объявленные в 2002 году. Технология сосредоточилась на проблеме управления цифровыми правами-ограничении распространения музыки и видео, защищенных авторским правом - но лежащие в основе механизмы вызвали интерес многих в сообществе MLS из-за потенциальных приложений MLS. Технология была запланирована для включения в будущий выпуск Windows под кодовым названием «Longhorn», но была исключена из планов Microsoft в 2004 году (Orlowski, 2004).

Возможно, несколько факторов способствовали отказу пространства продукта MLS. Microsoft ясно продемонстрировала, что существует гигантский рынок для продуктов, которые опускают MLS. Падение цен на компьютеры также сыграло свою роль: сайты, где пользователи обычно работают на нескольких разных уровнях безопасности, дешевле разместить два компьютера на каждом рабочем столе, чем пытаться развернуть продукты MLS. Наконец, цена и неопределенность совершенствования продукции MLS несомненно обескураживают много поставщиков. Трудно оправдать усилия по разработке «высокозащищенной» системы, когда вполне вероятно, что система все еще будет иметь идентифицируемые слабые места, такие как скрытые каналы, после всей дорогостоящей, специализированной работы.

1.9 Многоуровневая сеть

По мере снижения компьютерных расходов и повышения производительности в 1980-х и 1990-х годах компьютерные сети стали играть важную роль в совместном использовании работы и ресурсов. Задолго до того, как компьютеры регулярно подключались к Интернету, сайты создавали локальные сети для совместного использования принтеров и файлов. В оборонном сообществе многоуровневый обмен данными должен решаться в сетевой среде. Первоначально сообщество использовало сети дешевых

компьютеров как способ временно обойти проблему MLS. Вместо того, чтобы решать проблему обмена данными, многие организации просто развернули отдельные сети для работы на разных уровнях безопасности, каждый из которых работает в режиме высокой системы.

Такой подход не помог разведывательному сообществу. Многим проектам и департаментам необходимо обрабатывать информацию, содержащую различные разделы и кодовые слова. Просто было не практично предоставлять отдельные сети для каждой возможной комбинации отсеков и кодовых слов, так как их было так много. Кроме того, аналитики разведки часто тратили свое время на объединение информации из разных отделов для подготовки документа с другой классификацией. На практике эта работа требовала рабочего стола MLS и часто требовала связи по сети MLS.

Таким образом, в 1990-х годах сеть MLS приняла два разных пути. Организации в разведывательном сообществе продолжали преследовать продукты MLS. Это отражало потребности аналитиков разведки. В сети это вызвало помеченные сети, то есть сети, которые несли метки классификации на своем трафике, чтобы гарантировать, что ограничения MLS были применены.

Однако многие другие военные организации пошли другим путем. Компьютеры в большинстве военных организаций, как правило, группируются в сети, обрабатывающие данные до определенного уровня безопасности, работающие в режиме высокой системы. Этот выбор был обусловлен не архитектурным видением, а скорее влиянием архитектуры настольных сетей, появившейся на коммерческом рынке в сочетании с существующими стратегиями военной компьютерной безопасности. В конечном счете, эта стратегия была названа множественными одиночными уровнями (MSL) или множественными независимыми уровнями безопасности (MILS).

Маркированные сети.

Основной целью сети с маркировкой является предотвращение утечки секретной информации. Утечка может произойти из-за подслушивания в сетевой инфраструктуре или доставки данных в неустановленное место назначения. Это дало два разных подхода к маркированной сети. Более сложный подход используют криптографию для хранения различных уровней безопасности и предотвратить подслушивание. Более простой подход вставил метки безопасности в сетевой трафик и полагался на механизм эталонного монитора, установленный в сетевых интерфейсах для ограничения доставки сообщений.

На практике криптографическое оборудование и процессы управления ключами часто были слишком дорогими для использования в некоторых крупномасштабных сетевых приложениях MLS. Вместо этого сайты полагались на физическую безопасность для защиты своих сетей MLS от подслушивания. Это особенно верно в отношении разведывательного сообщества, где распространение отсеков и кодовых слов сделало

непрактичным использование криптографии для разделения уровней безопасности.

Внутри таких сайтов сетевая инфраструктура физически изолирована от любого контакта, за исключением людей с сверхсекретными разрешениями, поддерживаемыми специальными фоновыми исследованиями. Сетевые провода защищены от взлома, хотя и не от изощренных атак, которые могут соблазнить посторонних. Ограничения доступа MLS зависят от меток безопасности, встроенных в сетевые сообщения. Если криптография используется вообще, ее основная цель состоит в том, чтобы защитить целостность меток безопасности.

Стандартный трафик, использующий Интернет-протокол (IP), не включает метки безопасности, но Интернет-сообщество разработало стандарты для таких меток, начиная с IP Security Option (IPSO) (St.Johns, 1988). Разведывательное управление обороны США развило это дальше при реализации протокола для информационной системы разведки МО (DODIIS). Протокол, называемый сетью DODIIS безопасности для обмена информацией (DNSIX), указанных в маркировке и проверке, чтобы быть использован при передаче трафика через сетевой интерфейс DNSIX (LaPadula, Лемуан, Vukelich, и Вудворд, 1990). Для повышения надежности результирующей системы в спецификацию было включено описание проекта процесса проверки; проект был сверен с моделью безопасности, описывающей требуемое применение MLS.

В США криптографические методы MLS были исключительно областью агентства национальной безопасности (NSA), так как он установил стандарты для шифрования секретной информации. Традиционные протоколы NSA шифровали трафик на уровне канала, неся трафик без меток безопасности. В 1980-х и 1990-х годах АНБ запустило серию программ по разработке криптографических протоколов для обработки маркированных многоуровневых данных, включая систему защищенной сети передачи данных (SDNS), программу многоуровневой сетевой системы безопасности (MNSSP) и инициативу по многоуровневой Информационной системе безопасности (MISSI).

Этипрограммыдали Security Protocol 3 (SP3) и Message Security Protocol (MSP). SP3 защищает сообщения на уровне сетевого протокола (уровень 3) и был использован в устройствах шифрования шлюза для секретной сети маршрутизатора IP DOD (SIPRNET), которая разделяет секретную информацию на секретном уровне среди утвержденных военных и оборонных организаций. Тем не менее, SP3 является относительно старым протоколом и, вероятно, будет заменен вариантом протокола IP Security Protocol (IPSEC), который был адаптирован для оборонного сообщества, называемым High Assurance IP Interface Specification (HAIPIS). MSP защищает сообщения на уровне приложения и изначально был разработан для шифрования электронной почты. Система сообщений защиты, развивающаяся безопасная система электронной почты Министерства обороны, использует MSP.

Очевидно, что сеть MLS использует шифрование для защиты трафика от подслушивания. Кроме того, протоколы MLS могут использовать криптографию для применения ограничений доступа MLS. Протокол FIREFLY иллюстрирует это. Разработанный NSA для многоуровневых телефонных и сетевых протоколов, FIREFLY использует криптографию с открытым ключом для согласования ключей шифрования для защиты трафика между двумя объектами. Каждый сертификат FIREFLY содержит уровень доступа его владельца, и протокол сравнивает уровни при согласовании ключей. Если эти две организации используют защищенные телефоны, то протокол выдает ключ, уровень доступа к которому соответствует нижнему из двух организаций. Если два объекта устанавливают подключение к компьютерной сети, то согласование завершается успешно, только если уровни разрешения совпадают.

Множественные независимые уровни безопасности.

Несмотря на нехватку продуктов MLS, оборонные и разведывательные сообщества значительно расширили использование компьютерных систем в 1980-х и 1990-х годах. Вместо реализации систем MLS большинство организаций решили развернуть несколько компьютерных сетей, каждая из которых выделена для необходимого уровня безопасности. Это устранило риски совместного использования многопользовательских данных за счет устранения совместного использования многопользовательских данных на нескольких уровнях безопасности. При необходимости менее классифицированные данные копировались в одну сторону на серверы в сети более высокого уровня со съемного диска.

Для упрощения обмена данными в среде MILS многие организации внедрили устройства для передачи данных из сетей одного уровня безопасности в сети других уровней. Эти устройства обычно делятся на три категории:

- многоуровневые серверы-сетевые серверы, реализованные в системе MLS с отдельными сетевыми интерфейсами для подключения к высоким сетям системы, работающим на разных уровнях безопасности;
- односторонняя защита - устройства, которые могут передавать данные из сети на «низком» уровне безопасности в сеть на «высоком» уровне;
- понижение уровня защиты-устройства, которые могут передавать данные в любом направлении между сетями на разных уровнях безопасности.

На многоуровневом сервере компьютеры в сети с более низким уровнем безопасности могут хранить информацию на сервере, а компьютеры в сетях более высокого уровня могут посещать один и тот же сервер и извлекать эту информацию. Поставщики предоставляют множество многоуровневых серверов, включая веб-серверы, серверы баз данных и файловые серверы. В то время как такие системы популярны у некоторых оборонных организаций, другие избегают их. Большинство серверных продуктов достигают относительно низкого уровня надежности, что предполагает, что

злоумышленники могут найти способы утечки информации через них из более высокой сети в более низкую.

Односторонняя защита реализует одностороннюю передачу данных из сети на более низком уровне безопасности в сеть на более высоком уровне. Самые простые реализации полагаются на аппаратные ограничения, чтобы гарантировать, что потоки трафика только в одном направлении. Например, обычное сетевое оборудование оптического волокна поддерживает двунаправленное движение, но не трудно построить оборудование оптического волокна которое только содержит передатчик на одном конце и приемник на другом. Такое устройство может передавать данные в одном направлении без риска утечки данных в другой. Очевидным недостатком является то, что нет эффективного способа предотвратить перегрузку, так как низкая сторона не имеет возможности узнать, когда и были ли получены ее сообщения. Более изощренные приборы как NRLPump (Kang, Moskowitz, и Lee, 1996) во избежание этой проблемы путем снабжении подтверждения, используя доверенное программное обеспечение. Однако такие устройства могут страдать от того же недостатка, что и серверы MLS: существует очень мало надежных операционных систем, в которых можно реализовать надежное программное обеспечение MLS, и большинство из них достигают относительно низкой гарантии. Надежность часто ограничивается гарантией базовой операционной системы.

Понижение защиты важны, поскольку они затрагивают неприятный побочный эффект вычислений MLS: пользователи часто в конечном итоге заканчивают с сверхсекретными данными. Пользователь в секретной системе может работать как с конфиденциальными, так и с секретными файлами, и просто делиться этими файлами с другими пользователями в секретных системах. Однако он сталкивается с проблемой, если ему нужно предоставить конфиденциальный файл пользователю в конфиденциальной системе: как он предотвращает утечку секретной информации, когда он пытается предоставить чистую копию конфиденциального файла? Не существует простого и надежного способа сделать это, особенно при использовании коммерческих настольных компьютеров.

Та же проблема часто возникает в системах электронной почты: другой пользователь в сверхсекретной сети может пожелать отправить безобидное, но важное объявление своим коллегам по секретной сети. Он знает, что получатели имеют право получать содержимое сообщения, но как он гарантирует, что не будет утечки сверхсекретной информации вместе с сообщением электронной почты? Проблема также возникает в военных базах данных: база данных может содержать информацию на различных уровнях безопасности, но не все сообщества пользователей смогут обрабатывать данные на том же уровне, что и вся база данных. Чтобы быть полезными, данные должны быть очищены, а затем переданы пользователям на более низких уровнях классификации. Когда понижающий уровень защиты

освобождает информацию от более высокого уровня безопасности до более низкого, понижающий уровень обычно попадает в одну из трех категорий:

- ручной просмотр и выпуск-натренированный и доверенный оператор тщательно рассматривает все файлы, представленные на понижение. Если файл выглядит чистым от нечитаемой информации, то он освобождается;
- автоматический выпуск-система ищет индикаторы, чтобы показать, что уполномоченное лицо просмотрело файл и одобрило его для выпуска. Индикаторы обычно включают криптографическую аутентификацию, например цифровую подпись;
- автоматизированное рассмотрение-система использует набор тщательно построенных фильтров для того чтобы рассмотреть файл (архив), который нужно выпустить. Фильтры предназначены для проверки на наличие нечитаемых данных.

Традиционным методом был ручной анализ и выпуск. Сайт будет обучать оператора определять секретную информацию, которая не должна разглашаться, и оператор будет вручную просматривать все данные, проходящие через охрану. Эта стратегия оказалась непрактичной, поскольку стало очень трудно надежно сканировать файлы на наличие конфиденциальной информации. Текстовые процессоры, такие как Microsoft Word, как правило, сохраняют конфиденциальную информацию даже после того, как пользователь попытался удалить ее из файла (Byers, 2004). Другая проблема-это стеганография: пользователь, находящийся на высокой стороне защиты, или сложная часть программного обеспечения, может легко вставлять большие элементы данных в графические изображения или другие цифровые файлы, чтобы визуальный обзор не обнаружил их присутствие. В дополнение к проблеме надежного сканирования, существует проблема человеческого фактора: немногие операторы будут оставаться эффективными в этой работе в течение очень долгого времени. Офицеры военной безопасности рассказывают о том, что операторы обзора попадают в режим, в котором они автоматически одобряют все без обзора, частично для поддержания пропускной способности сообщений и частично из-за скуки.

Подход к автоматизированному выпуску использовался стандартной службой Mail Guard (SMG) (Smith, 1994). SMG приняла текстовые сообщения электронной почты, которые были проверены автором сообщения, явно помечены для выпуска и подписаны цифровой подписью с использованием протокола DMS. SMG будет проверять цифровую подпись и идентификацию подписывающего лица по списку пользователей, уполномоченных отправлять электронную почту через охрану. Группа SMG будет также искать в сообщении слова, связанные с секретной информацией, которые не должны разглашаться, и блокировать сообщения, содержащие такие слова. Авторизованные пользователи также могут передавать файлы, прикрепляя их к электронной почте. Прикрепленные файлы должны были быть проверены, а затем «запечатаны» с помощью специальной прикладной программы: SMG проверит наличие печати, прежде чем выпустить прикрепленный файл.

С 1980-х годов некоторыезащиты используют автоматизированный подход к обзору, главным образом для выпуска записей из баз данных с высокой степенью секретности в сети с более низким уровнем безопасности. Многие из этих защит были разработаны для автоматического просмотра данных развертывания сил в высоком формате. Они были настроены с подробными правилами о том, как проверить поля записей базы данных, так что данные были освобождены на правильном уровне безопасности. В некоторых случаях давали указания о том, как очистить определенные поля базы данных, чтобы удалить строго засекреченные данные перед выпуском записей.

В то время как защита и многоуровневые серверы обеспечивают явное преимущество, позволяя обмен данными между системами высоких сетей, работающих на различных уровнях очистки, они также создают проблемы. Самая очевидная проблема заключается в том, что он помещает все яйца MLS в одну корзину: охранник централизует защиту MLS в одном устройстве, что, несомненно, привлекает интерес злоумышленников. Понижение уровня защиты вызывает особую озабоченность, поскольку существует много способов, которыми троянский конь на «высокой» стороне безопасности может замаскировать конфиденциальную информацию, чтобы она успешно проходила через понижающие фильтры охраны. Например, троянец может встроить секретную информацию в явно несекретное изображение с помощью стеганографии. Другая проблема заключается в том, что самое безопасное место для прикрепления метки к данным находится в точке происхождения данных: защита с меньшей вероятностью правильно маркируют данные, поскольку они удаляются из точки происхождения данных (Saydjari, 2004). Безопасности, которые использовали автоматизированный механизм выпуска, могут быть несколько менее склонны к этой проблеме, если безопасность основывает свое решение на криптографически-защищенной метке, предоставленной в точке происхождения данных. Однако это преимущество может быть компенсировано другими рисками, если защита или процесс маркировки размещены в операционной системе с низким уровнем безопасности.

2Архитектуры многоуровневой защиты реляционной системы управления базами данных

Архитектуры многоуровневой защиты реляционной системы управления базами данных (РСУБД) можно разделить на два основных типа в зависимости от того, применяется ли обязательное управление доступом самой СУБД или делегируется доверенной операционной системе. Эти два общих типа являются архитектурой Woods Hole и архитектурой Trusted Subjects (доверенных субъектов).

2.1 АрхитектураWOODSHOLE

Архитектура Woods Hole является результатом трехнедельного исследования по доверенному управлению данными, спонсируемого ВВС США в Вудс-Хоул, штат Массачусетс, в 1982 году. Предметом данного исследования было следующее: можем ли мы построить многоуровневую безопасную СУБД с использованием существующих ненадежных готовых СУБД с минимальными изменениями?

Архитектуры Woods Hole предполагают, что для доступа к данным используется ненадежная готовая СУБД, а доверенный код разрабатывается вокруг этой СУБД для обеспечения общей защищенной СУБД. Они могут быть разделены на две основные категории: архитектур с ядром и распределенных архитектур.

2.1.1 Архитектура с ядром

Архитектура с ядром использует доверенную операционную систему и несколько копий готовых СУБД, где каждая копия связана с некоторым доверенным интерфейсом. Каждая пара (доверенный интерфейс, СУБД) связана с определенным уровнем безопасности. Доверенная операционная система применяет политику полного контроля доступа ко всем доступам к объектам РСУБД с помощью РСУБД. Это гарантирует, что данные на разных уровнях безопасности хранятся отдельно и что каждая копия СУБД получает доступ к данным, авторизованным для соответствующего уровня безопасности. Последнее возможно, поскольку многоуровневая база данных разбивается на несколько одноуровневых баз данных, каждая из которых представляет собой фрагмент концептуальной многоуровневой базы данных. Каждый фрагмент хранится в одноуровневом объекте операционной системы (например, в файле), который помечен операционной системой на соответствующем уровне безопасности и, таким образом, может быть доступен только в соответствии с политикой обязательного контроля доступа (mandatoryaccesscontrol, MAC) операционной системы.

На рисунке 2.1 показана архитектура с ядром, где одна РСУБД связана с уровнем безопасности «High», а другая РСУБД связана с уровнем

безопасности «Low». РСУБД, связанные с уровнем безопасности «High», имеют доступ как к фрагменту базы данных с высоким уровнем безопасности, так и к фрагменту базы данных с низким уровнем безопасности. Но СУБД, связанные с уровнем безопасности «Low», имеют доступ только к фрагменту базы данных с низким уровнем безопасности.

Преимуществом этой архитектуры является то, что данные на разных уровнях безопасности изолированы в базе данных, что позволяет обеспечить более высокий уровень безопасности. Еще одно преимущество заключается в том, что при условии, что операционная система уже оценена, эта архитектура должна свести к минимуму количество времени и усилий для оценки РСУБД. Однако эта архитектура приводит к дополнительным накладным расходам, так как доверенной операционной системе необходимо разделять данные на разных уровнях безопасности при добавлении в базу данных, а также может потребоваться объединение данных с разных уровней безопасности при получении данных копией РСУБД, связанной с высоким уровнем безопасности.

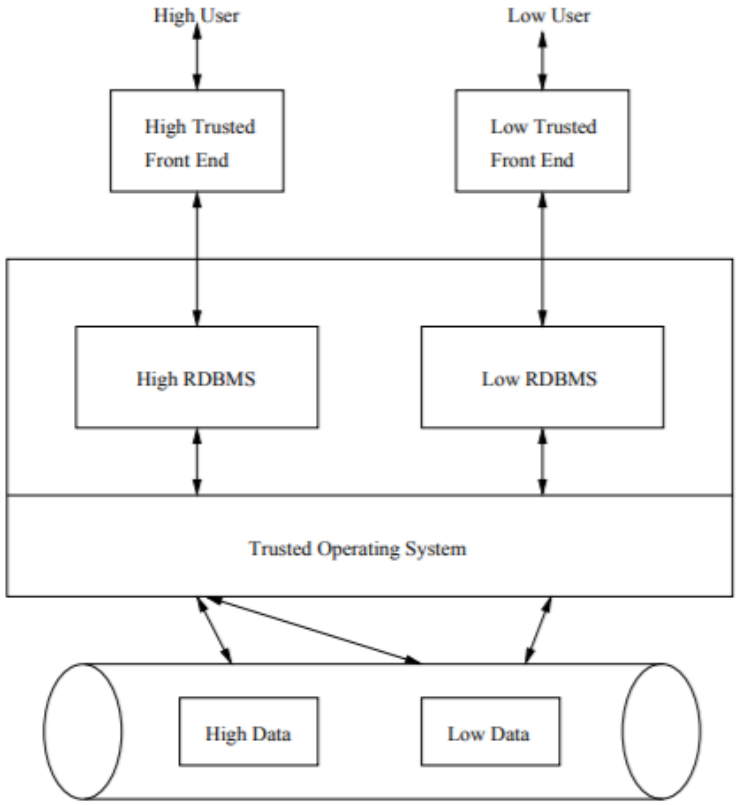

Рисунок 2.1 – Архитектура многоуровневой защиты РСУБД с ядром

2.1.2 Распределенная архитектура

Распределенная архитектура является разновидностью архитектуры с ядром. Данная архитектра использует несколько копий доверенного интерфейса и РСУБД, каждая из которых связана с собственным хранилищем

базы данных. В этой архитектурной схеме РСУБД уровня безопасности 1 содержит реплику каждого элемента данных, к которому субъект уровня 1 может получить доступ. Таким образом, при извлечении данных РСУБД извлекает их только из собственной базы данных. Другое преимущество этой архитектуры заключается в том, что данные физически разделены на отдельные аппаратные базы данных. Однако эта схема приводит к дополнительным накладным расходам при обновлении данных, поскольку различные реплики необходимо синхронизировать.

2.2 АрхитектураTRUSTEDSUBJECTS

Архитектура доверенного субъекта (trustedsubject) - это схема, содержащая доверенную РСУБД и доверенную операционную систему. В соответствии с этой архитектурой обязательная политика управления доступом применяется самой РСУБД. Объекты базы данных (например, таблица) хранятся в объектах операционной системы (например, в файле), помеченных на самом высоком уровне безопасности. Таблица базы данных может содержать строки с различными уровнями безопасности. Такие строки различаются на основе их уровня безопасности, который явно хранится с каждой строкой. Эта архитектура называется «доверенным субъектом», поскольку РСУБД имеет право нарушать политику MAC операционной системы при доступе к объектам базы данных. Например, когда пользователь с низким уровнем безопасности запрашивает таблицу базы данных, объект операционной системы, в котором хранится эта Таблица, заканчивается доступом, что является нарушением политики MAC операционной системы. Но РСУБД доверяют возвращать пользователям только те строки, для которых он авторизован в соответствии с политикой MAC. На рисунке2.2 показана многоуровневая безопасная архитектура СУБД доверенных субъектов.

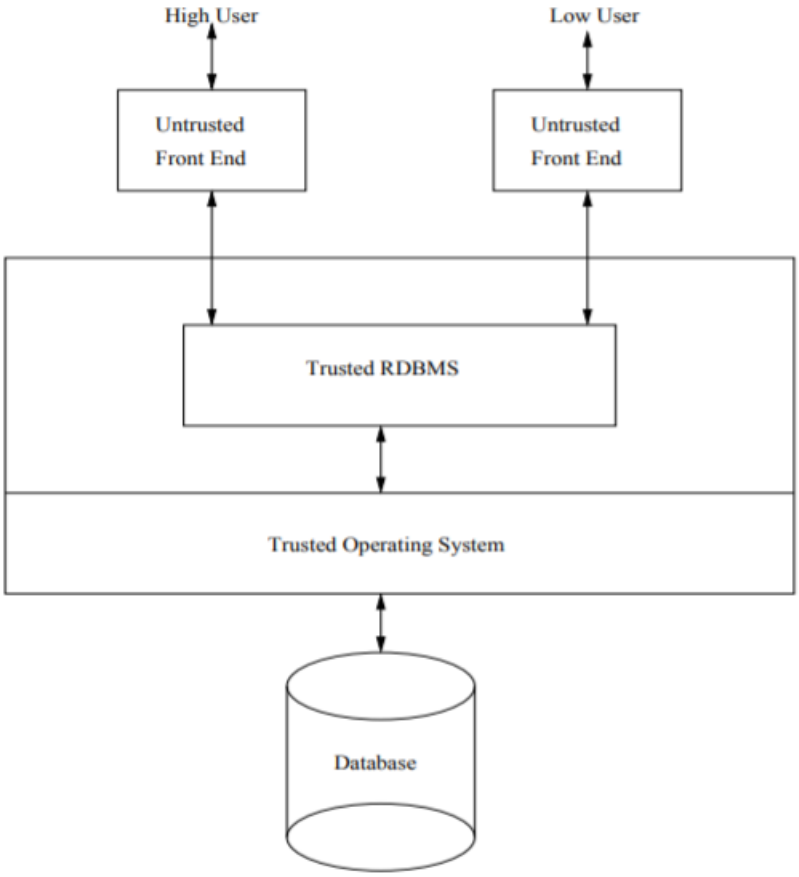

Рисунок 2.2 – Архитектура многоуровневой защиты РСУБД доверенного субъекта

Преимуществом этой архитектуры является то, что РСУБД имеет доступ ко всем уровням данных одновременно, что сводит к минимуму извлечение и обработку обновлений. Однако эта архитектура приводит к созданию специальных РСУБД, для которых требуется разработка и проверка большого количества доверенного кода наряду с обычными функциями РСУБД. Он также не имеет потенциала для оценки на высоких классах оценки TCSEC (Trusted Computer System Evaluation Criteria), потому что встреча более высоких уровней гарантии требует способности обеспечить разделение обязательных объектов некоторой формой аппаратного решения. Также трудно доказать, что доверенное программное обеспечение, используемое для изоляции обязательных объектов (например, строк данных с различными уровнями безопасности), работает правильно, не допуская потока данных с высоким уровнем безопасности для пользователей с низким уровнем безопасности.

3Общая модель для оценки многоуровневой безопасности объектно-ориентированных программ

Многие исследователи предложили различные подходы, которые помогают в разработке более безопасных программ. Один из этих подходов определяет перечень принципов, которые могут быть использованы в качестве руководства при разработке защищенных систем. Другие подходы разработали общие принципы кодирования, направленные на выявление уязвимостей кода. Однако эти подходы не могут обнаружить риски безопасности на ранней стадии и не способны количественно оценить безопасность данных программ.

С другой стороны, многоуровневая безопасность является важнейшим аспектом информационных систем. На самом деле, многие признают, что это является обязательным для любой системы, чтобы иметь различные уровни безопасности. Другими словами, различные критически важные для безопасности компоненты любой системы должны быть классифицированы на различные уровни критичности с точки зрения информационной безопасности. Существует несколько моделей безопасности для классификации информации с учетом ее чувствительности к безопасности. Например, модель США определяет четыре уровня классификации безопасности (сверхсекретная, секретная, конфиденциальная и несекретная). Другой пример - британцы, которые используют модель шести уровней классификаций безопасности (сверхсекретных, секретных, конфиденциальных, ограниченных, защищенных и несекретных).

К сожалению, большинство исследований по метрикам безопасности программного обеспечения сосредоточены на определении метрик путем классификации компонентов системы на классифицированные и не классифицированные. Кроме того, существуют некоторые исследования, которые изучают многоуровневую безопасность, такие как работа Котенко и Дойникова, которая определяет метрики безопасности для различных уровней системы. Однако эта работа не учитывает поток информации о различных значениях безопасности критически важных данных. Вместо этого он фокусируется на разделении различных типов метрик для разных частей системы.

Ни один из существующих проектов не разработал метрики безопасности программы на основе на артефактах проектирования, учитывающих различный уровень безопасности компонентов системы. В данной главе мы определяем общую модель, которая применима к любой критически важной для безопасности объектно-ориентированной программе для оценки ее безопасности в отношении потенциального потока критически важных для защиты данных.

3.1 Методология исследования

Подход Устстона и Монни гласит, что реализация любого проекта должна осуществляться в два этапа. Первая-аналитическая часть, посвященная определению теории проекта. Вторая называется эмпирической частью, которая проводится для доказательства аналитической части.

Итак, первое: целью является определение модели, способной оценивать защищенность объектно-ориентированных программ с многоуровневыми классификациями защиты данных. Модель концентрируется на конкретных объектно-ориентированных свойствах проекта, которые оказывают наибольшее влияние на безопасность программ. Предлагаемые метрики должны измерять безопасность программ путем определения потенциального информационного потока критически важных для безопасности данных.

Второе: в главе будет определен ряд метрик безопасности для объектно-ориентированных программ с многоуровневыми классификациями безопасности их компонентов. Разработанные метрики должны соответствовать определенным принципам проектирования безопасности для разработки безопасных программ. Эти метрики могут быть применены к любой объектно-ориентированной программе, если она имеет схему классов. Такие метрики помогут разработчикам программного обеспечения оценить уровень безопасности своих программ с ранней стадии разработки на основе артефактов проектирования программ.

3.2 Допущения модели

Модель направлена на введение набора показателей безопасности для программ с многоуровневыми классификациями секретности данных на основе их конструкций. Эти метрики измеряют потенциальный поток критически важных для безопасности данных данного объектно-ориентированного проекта. Каждая метрика определяется в соответствии с определенным принципом проектирования безопасности, которого необходимо придерживаться для достижения безопасной программы. Модель фокусируется на определении таких метрик по четырем объектно-ориентированным свойствам (т. е. инкапсуляции данных, связности, связи и размеру проекта). Эти метрики являются сравнительным измерением, которое означает, что результаты этих метрик могут быть использованы для сравнения относительной безопасности различных объектно-ориентированных программ в отношении этих четырех свойств.

Метрики были разработаны таким образом, чтобы их результаты находились в диапазоне от 0 до 1. Чем меньше значение показателя, тем безопаснее программа. Аналогично, по мере увеличения значения метрики программа становится менее защищенной. Это означает, что чем ниже значения этих метрик, тем больше они придерживаются связанных с ними принципов проектирования безопасности. И более высокие значения

означают, что они более или менее придерживаются к их уместным принципам проектирования безопасностью.

Один из подходов, который может быть использован для применения метрик безопасности, основанных на проекте программы, разработан Алшамари. Этот подход зависит от точного предоставления аннотированных классов критически важных для безопасности данных для данной объектно-ориентированной программы с помощью аннотаций UMLsec и SPARKS. В этом подходе аннотации UMLsec используются для аннотирования атрибутов, методов и классов с «секретностью», если они содержат или взаимодействуют с критически важными для безопасности данными. Кроме того, заметки SPARKS используются для отображения взаимодействия методов с критически важными для безопасности атрибутами, методами и классами.

Вместо этого эта модель развивает новый подход, который помогает в применении метрик проектирования безопасности, определенных здесь. Этот подход опирается на разработчиков программы, чтобы обеспечить точно разработанную двумерную матрицу. Эта матрица показывает уровень безопасности всех взаимодействий с критически важными для безопасности классами, которые вызваны методами или классами в данной конструкции.

Таблица 3.1 – Терминология модели

Название	Описание
Критический с точки зрения безопасности атрибут	Атрибут, содержащий конфиденциальные данные
Критический с точки зрения безопасности метод	Функция, которая обращается или взаимодействует с критическими для безопасности атрибутами
Критический с точки зрения безопасности класс	Класс, имеющий, по крайней мере, один критический для безопасности атрибут или один критический для безопасности метод

Чтобы проиллюстрировать этот подход, класс извлекается из определенной программы, называемой информационной системой студента, показанной на рисунке 3.1, и матрица, связанная с этим классом, показана на рисунке 3.2. Предположим, что модель безопасности, используемая в этом контексте это {0,1,2,3}; при этом 0 не является критическим для безопасности атрибутом, а 3 является наиболее критичным для безопасности атрибутом.В этом классе предположим, что name не является критическим с точки зрения безопасности атрибутом (следовательно, его значение безопасности равно 0), и доступ к нему осуществляется методом getName. DOB должен быть критическим с точки зрения безопасности атрибутом со значением безопасности 1 и доступен только методом getDOB. Также предположим, что nationalID является еще одним критическим с точки зрения безопасности атрибутом со значением безопасности 2 и доступен только методом

getNationalID. Предположим, что portalPassword также является критическим с точки зрения безопасности атрибутом со значением безопасности 3 и доступен только методом getPortalPassword. Затем матрицу безопасности, связанную с диаграммой классов из рисунка 3.1, можно увидеть на рисунке 3.2.

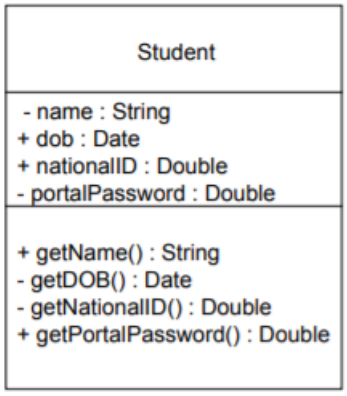

Рисунок 3.1– Диаграмма класса Студент

		Attributes			
		- name	+ dob	+ nationalID	- portalPassword
Methods	+ getName()	0	0	0	0
	- getDOB()	0	1	0	0
	- getNationalID()	0	0	2	0
	+ getPortalPassword()	0	0	0	3

Рисунок 3.2–Матрица безопасности класса Студент

3.3 Принципы проектирования безопасности

В этом разделе будут рассмотрены принципы проектирования системы безопасности, относящиеся к метрикам проектирования системы безопасности, определенным в этой модели. Нами были выбраны два принципа проектирования, которые будут изучены при разработке этой модели (т. е. наименьшая привилегия и уменьшение поверхности атаки). Они были выбраны на основе того, что именно эти два принципа могут оказать наибольшее влияние на разработку безопасных систем, и, следовательно, они должны быть интенсивно рассмотрены при определении метрик безопасности. Таким образом, эти метрики построены с учетом этих двух принципов для измерения безопасности конструкций с точки зрения потенциального потока информации.

Принцип наименьшей привелегии.

Этот принцип описывается как разрешение программам и пользователям выполнять определенную работу с наименьшими привилегиями. Его основные преимущества заключаются в минимизации взаимодействия между привилегированными компонентами в данной системе и, следовательно, минимизации потерь в случае успешной атаки.

Принцип уменьшения поверхности атаки.

Этот принцип также направлен на то, чтобы сделать программы более безопасными за счет уменьшения количества компонентов, которые могут быть достигнуты извне системы. Существует несколько подходов к сокращению таких компонентов. Общий подход, описанныйГовардом, предлагает уменьшить количество работающего кода, отключив ненужные функции системы. Другой подход заключается в минимизации количества точек входа в систему, к которым могут получить доступ недоверенные пользователи.

3.4 Модель оценки многоуровневой защиты

Из-за важности наличия нескольких уровней безопасности данных для любых систем, определяется модель, которая способна определить набор метрик безопасности, которые отвечают этой цели. Эта модель учитывает четыре наиболее распространенных свойства проектирования программного обеспечения для определения этих метрик. Данные свойства включают: инкапсуляцию данных, сцепление, связь и размер проектирования систем. Эти показатели зависят от уровней безопасности, определенных аналитиками безопасности системы. Кроме того, модель разрабатывает матрицу безопасности, которую можно легко использовать для извлечения требуемой информации из этих метрик проектирования безопасности.

Эти метрики проектирования безопасности определены как универсальные, что означает, что они могут быть применены к любой программе независимо от типа модели безопасности, которую она использует. Для иллюстрации рассмотрим набор значений модели безопасности, используемой в данном контексте, как V = {0,..., V}, где атрибут, обозначенный как 0 означает, что это не критически важный для безопасности и атрибута обозначены как V (максимальное число в модели) означает, что он является наиболее критическим с точки зрения безопасности атрибута в этой программе (пусть оператор величины $|S|$ возвращает размер заданного множества S). Эти метрики определяются следующим образом.

Доступностькритическихдлябезопасностиатрибутов(AccessibilityofSe curity-CriticalAttributesMetric, ASCAM).

Свойство инкапсуляции данных в объектно-ориентированных конструкциях (проектах) связано с наличием ограничений на доступность данных внутри данной программы. С точки зрения безопасности, было показано, что это свойство оказывает существенное влияние на безопасность программы, включая работу Маруяма и Алшаммари. Эти исследования

показали, что создание критических для безопасности атрибутов, менее доступных извне их классов, делает программы более безопасными. Это в конечном итоге удовлетворит спецификациям принципа безопасности «уменьшение размера поверхности атаки».

Эта метрика предназначена для измерения доли критических атрибутов безопасности, которые доступны из-за пределов их класса в данной конструкции для объектно-ориентированной программы по отношению к их уровням безопасности. Таким образом, эта метрика определяется как; «отношение числа критичности некритических атрибутов безопасности в данном проекте к общему числу критичности критических атрибутов безопасности в этом проекте».

РассмотриммножествоатрибутоввосновепроектаDкакA_i, $i \in \{1,...,a\}$,множествокритическихдлябезопасностиатрибутоввDкакSCAj, $j \in \{1,...,sca\}$ такие, что$SCA \subseteq A$, и множество неприватных (доступных) критических для безопасности вDкак$ASCA_k$, $k \in \{1,...,asca\}$ таких, что$ASCA \subseteq SCA \subseteq A$. Тогда, ASCAMвыражается:

$$ASCAM(D) = \frac{\sum_{k=1}^{asca}(ASCA_k \times v)}{|SCA| \times \max(V)} \quad (3.1)$$

Доступностькритическихдлябезопасностиметодов(Accessibility of Security-Critical Methods Metric, ASCMM).

Как было показано ранее, свойство инкапсуляции данных в объектно-ориентированных конструкциях связано с наличием ограничений на доступность данных внутри данной программы. Другим возможным способом доступа к данным являются методы, которые имеют доступ к атрибутам. Подобно прямой доступности критически важных для безопасности атрибутов, доступ к критически важным атрибутам безопасности косвенно с помощью методов, которые взаимодействуют с ними, может оказывать такое же влияние на безопасность. Поэтому рекомендуется иметь меньшую доступность к методам, которые взаимодействуют с критически важными для безопасности атрибутами, чтобы удовлетворить принципу проектирования безопасности «уменьшение размера поверхности атаки».

Этот показатель предназначен для измерения доли методов, которые имеют доступ или взаимодействие с критически важными для безопасности атрибутами и доступны из-за пределов их класса в данной объектно-ориентированной конструкции по отношению к их уровням безопасности. Таким образом, эта метрика определяется как; «отношение числа критичности некритических методов безопасности в данной конструкции к общему числу критичности критических методов безопасности в этой конструкции».

РассмотриммножествометодоввпроектеDкакM_i, $i \in \{1,...,m\}$, множествокритическихдлябезопасностиметодоввDкакSCMj, $j \in \{1,...,scm\}$ таких, что$SCM \subseteq M$, и множество неприватных (доступных) критических для безопасности методов вDкак$ASCM_k$, $k \in \{1,...,ascm\}$ таких,

чтоASCM⊆SCM⊆M (при заданном множестве S, пусть оператор величины |S| возвращает размер множества). Тогда ASCMMвыражается как:

$$ASCMM(D) = \frac{\sum_{k=1}^{ascm}(ASCM_k \times v)}{|SCM| \times \max(V)} \quad (3.2)$$

Согласованность критических для безопасности методов метрики(CohesivenessofSecurity-CriticalMethodsMetric, CSCMM)

С точки зрения объектно-ориентированных свойств проектирования рекомендуется иметь целостный проект, который увеличивает взаимодействие атрибутов с методами внутри их класса. Что касается безопасности, рекомендуется уменьшить согласованность взаимодействий между критически важными для безопасности атрибутами и методами внутри их классов, чтобы иметь более безопасные программы. Согласованность - это привилегии над критически важными для безопасности атрибутами, и чем меньше таких взаимодействий, тем больше это соответствует принципу наименьших привилегий.

Основная цель этой метрики состоит в том, чтобы измерить степень потенциального потока критически важных для безопасности данных, вызванного взаимодействием между критически важными для безопасности атрибутами и методами в данном объектно-ориентированном дизайне, принимая во внимание уровни безопасности этих атрибутов. Таким образом, эта метрика определяется как «отношение числа методов, взаимодействующих с критически важными для безопасности атрибутами для каждого класса, к максимальному числу методов, которые могут взаимодействовать с атрибутами для каждого класса в конкретной программе».

Для вычисления этой метрики число взаимодействий с каждым критичным для безопасности атрибутом умножается на уровень безопасности для каждого класса. Затем сумма этих значений делится на общее количество возможных способов взаимодействия с атрибутами в этом классе. Это может быть вычислено путем умножения общего числа методов в этом классе на общее число атрибутов в том же классе. Затем он умножается на максимальное значение модели безопасности, используемой в контексте. Значения связности всех классов суммируются, а затем делятся на количество классов в дизайне, чтобы принять среднюю связностьдизайна.

РассмотриммножествоклассоввпроектеDкакCi , i∈ {1,..., c}, множествометодоввтакойжеконструкцииMj , j∈ {1,...,m}, множествоатрибутоввDконструкциикакAk , k∈ {1,...,a}, имножествокритическихдлябезопасностивDкакSCA_l , l∈ {1,...,sca} таких, чтоSCA⊆A. Пусть α (SCAl) число методов, которые обращаются к критическим для безопасности атрибутамSCA_l . ТогдаCSCMMвыражается как:

$$CSCMM(D) = \frac{\sum_{i=1}^{c} \frac{\sum_{l=1}^{sca} \alpha(SCA_l) \times v}{|M| \times |SCA| \times \max(V)}}{|C|} \quad (3.3)$$

Связьметрикикритическихклассовбезопасности (Coupling of Security-Critical Classes Metric CSCCM).

Связь является одним из наиболее распространенных объектно-ориентированных свойств проектирования, которые были широко рассмотрены в нескольких исследованиях. Это свойство определяется количеством взаимодействий объекта с другими объектами внутри программы. Многие исследования показали, что для объектно-ориентированных программ рекомендуется использовать слабосвязанные компоненты, чтобы быть более многоразовыми, понятными и расширяемыми.

Что касается информационной безопасности, то было показано, что существует высокая корреляция между связью и незащищенностью программ. Исследование Лю и Траоре показало, что успешные атаки во многих случаях вызваны сильно связанными объектами. Кроме того, исследования Алшаммари показали, что слабосвязанные программы могут уменьшить потенциальный поток критически важной для безопасности информации и, таким образом, создать более безопасные программы. Это удовлетворяет принципу проектирования безопасности «наименьших привилегий».

Поэтому эта модель фокусируется на влиянии связи на безопасность программ и определяет метрику безопасности, которая соответствует описанной здесь многоуровневой модели безопасности. Эта метрика связи предназначена для измерения потенциального появления связей между критически важными для безопасности атрибутами и классами в заданном проекте с учетом уровня безопасности каждого соединения. Таким образом, этот показатель определяется как; «Отношение критичности числа ассоциаций всех классов с критическими для безопасности атрибутами к максимальному числу ассоциаций, которые могут возникнуть с критическими для безопасности атрибутами для каждого класса в конкретной программе».

РассмотриммножествоклассоввпроектеDкакCi , i∈ {1,..., c}, множествоатрибутоввпроектеDкакAk , k∈ {1,...,a}, имножествокритическихдлябезопасности атрибутов в DкакSCA_l, l∈ {1,...,sca} таких, что SCA⊆A. Пусть β(SCA_l) – числоклассов, связанных с критически важным для безопасности атрибутом SCA_l. ТогдаCSCCMвыражается как:

$$CSCCM(D) = \frac{\sum_{l=1}^{sca} \beta(SCA_l) \times v}{|C-1| \times |SCA| \times \max(V)} \quad (3.4)$$

Критический для безопасности метрический размер конструкции (Security-CriticalDesignSizeMetric, SCDSM).

Размер конструкции (проекта) -это одно из объектно-ориентированных свойств, которое необходимо учитывать на ранней стадии жизненного цикла разработки программы, поскольку оно оказывает существенное влияние на ее

повторное использование и функциональность. Благодаря этой важности Бансия и Дэвис определили показатель, который связан с этим свойством. Эта метрика называется размером проекта в классах (Design Size in Classes, DSC), который измеряет количество классов в конкретном проекте.

С точки зрения безопасности программ в ряде исследований изучено влияние свойства размера конструкции. Это включает в себя работу Чоудхори, которая определила метрику для измерения отношения критических сегментов кода в коде конкретной программы. Работа Алшаммари также изучила это свойство. Они показывают, что желательно иметь небольшой размер дизайна критических для безопасности классов, чтобы придерживаться принципа проектирования безопасности «уменьшение размера поверхности атаки».

Модель, определенная в этой статье, рассматривает важность свойства размера конструкции для безопасности и, следовательно, определяет метрику для этой цели. Основная цель этого показателя состоит в том, чтобы измерить долю критически важных для безопасности классов с учетом их уровней безопасности. Таким образом, эта метрика определяется как «отношение критичности числа критических классов безопасности к общему числу критических классов безопасности в структуре конкретной программы».

Рассмотрим множество классов в конструкции DкакCi , i∈ {1,..., c}, множествоатрибутоввконструкцииDкакAk , k∈ {1,...,a}, множествокритических для безопасности атрибутов вDкакSCA_l, l∈ {1,...,sca} таких, чтоSCA⊆A, и множество классов с критическими для безопасности классами, определенных в них в конструкции DкакSCC_l, l∈ {1,...,scc} таких, чтоSCC⊆C. ТогдаSCDSопределяется как:

$$CSDSM(D) = \frac{\sum_{l=1}^{scc}(SCC_l \times v)}{|C| \times \max(V)} \quad (3.5)$$

3.5 Исследование модели

В этом разделе будет показано, как применить метрики многоуровневого проектирования безопасности, определенные здесь для конкретной банковской системы. Для измерения безопасности данной программы в отношении этих метрик разработчикам системы необходимо предоставить полную схему классов UML. Также необходимо предоставить матрицу безопасности, определенную этой моделью, как объяснено в предыдущем разделе. Эта матрица показывает взаимодействия каждого атрибута с другими методами и классами, а также уровень безопасности каждого взаимодействия.

- Диаграмма классов UMLбанковской системы.

Диаграмма классов UML, показанная на рисунке 3.3, является частью банковской системы, извлеченной, чтобы показать, как измерить безопасность определенной программы в отношении модели, определенной в этой работе.

Эта схема состоит из классов, которые отвечают за хранение информации о клиентах определенного банка. Диаграмма классов состоит из пяти классов: клиенты, кредитные карты, дебетовые карты, кредиты и чеки. Каждый из этих классов отвечает за конкретную задачу и содержит ряд атрибутов и методов для серфинга с этой целью.

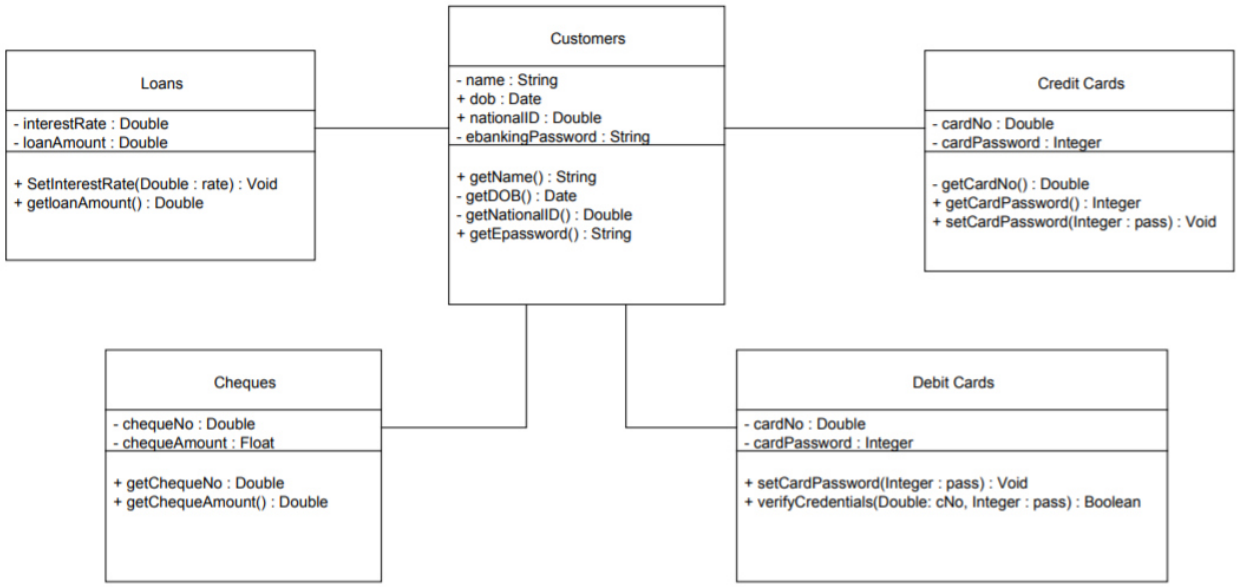

Рисунок 3.3- Диаграмма классов UMLбанковской системы

Например, класс Customers отвечает за хранение личной информации о клиентах, такой как их имя, дата рождения, национальный идентификатор и пароль ebanking. Он также определяет ряд методов, связанных с получением определенных данных в ответ на запросы внутри или вне класса. Класс Loans хранит информацию о кредитах клиента, например сумму кредита. Он также выполняет функции, которые отвечают за возвращение этой информации после запроса. Класс Cheques предоставляет сведения о чеках клиента и определяет функции, которые получают доступ к такой информации. Классы кредитных и дебетовых карт определяют атрибуты, в которых хранятся номера карт и их пароли. Они также содержат методы, возвращающие и обновляющие эту информацию.

- Матрица банковской системы.

Данная матрица показывает взаимодействие между атрибутами и методами во всей системе. Она также показывает те классы, которые непосредственно обращаются к определенным атрибутам извне их классов. Эта матрица безопасности должна включать значение безопасности для каждого взаимодействия. Это значение совпадает с уровнем безопасности атрибута, с которым произошло взаимодействие.

Для целей данного примера предположим, что модель безопасности, используемая в данном контексте, {0,1,2,3,4}. Атрибуты, значение безопасности которых равно 0, указывают на то, что они не являются

критически важными для безопасности. Кроме того, атрибуты являются критически важными для безопасности, и их уровень критичности зависит от их значения безопасности. Это означает, что при увеличении этого значения критичность безопасности определенного атрибута возрастает. Например, атрибуты, которым присвоено значение безопасности 1, указывают, что они имеют наименее важные для безопасности данные, в то время как атрибуты, которым присвоено значение безопасности 4, показывают наиболее важные для безопасности данные. Кроме того, матрица показывает доступность каждого атрибута, метода и класса в системе. Эта информация извлекается из модификаторов доступа, показанных на схеме классов UML системы.

Таблица3.2 – Уровни безопасности атрибутов банковской системы

Класс	Атрибут	Уровень безопасности
Customers	Name	0
Customers	Dob	2
Customers	NationalID	3
Customers	PortalPassword	4
Credit Cards	CardNo	4
Credit Cards	CardPassword	4
Debit Cards	CardNo	3
Debit Cards	CardPassword	4
Cheques	ChequeNo	1
Cheques	ChequeAmount	2
Loans	InterestRate	0
Loans	LoanAmount	1

На схеме классов UML на рисунке 3.3 представлено пять классов, каждый из которых имеет ряд атрибутов с различными уровнями безопасности. Предположим, что системный аналитик предоставил таблицу 3.2, которая показывает уровни безопасности для всех атрибутов в системе. Эти значения используются для построения матрицы безопасности банковской системы, как показано на рисунке 3.4.

		Attributes											
		- Customers. name (0)	+ Customers. dob (2)	+ Customers. nationalID (3)	- Customers. portalPassword (4)	- CreditCards. cardNo (4)	- CreditCards. cardPassword (4)	- DebitCards. cardNo (3)	- DebitCards. cardPassword (4)	- Cheques. chequeNo (1)	- Cheques. chequeAmount (2)	- Loans. interestRate (0)	- Loans. loanAmount (1)
Classes	+ Customers	0	0	3	0	0	0	0	0	0	0	0	0
	+ Credit Cards	0	2	3	0	0	0	0	0	0	0	0	0
	+ Debit Cards	0	2	3	0	0	0	0	0	0	0	0	0
	+ Cheques	0	0	3	0	0	0	0	0	0	0	0	0
	+ Loans	0	0	3	0	0	0	0	0	0	0	0	0
Methods	+ Customer.getName	0	0	0	0	0	0	0	0	0	0	0	0
	- Customer.getDOB	0	2	0	0	0	0	0	0	0	0	0	0
	- Customer.getNationalID	0	0	3	0	0	0	0	0	0	0	0	0
	+ Customer.getPortalPassword	0	0	0	4	0	0	0	0	0	0	0	0
	- CreditCards.getCardNo	0	0	0	0	4	0	0	0	0	0	0	0
	+ CreditCards.getCardPassword	0	0	0	0	0	4	0	0	0	0	0	0
	+ CreditCards.setCardPassword	0	0	0	4	0	4	0	0	0	0	0	0
	+ DebitCards.setCardPassword	0	0	0	0	0	0	0	4	0	0	0	0
	+ DebitCards.verifyCredentials	0	0	0	4	0	0	3	4	0	0	0	0
	+ Cheques.getChequeNo	0	0	0	0	0	0	0	0	1	0	0	0
	+ Cheques.getChequeAmount	0	0	0	0	0	0	0	0	0	2	0	0
	+ Loans.setInterestRate	0	0	0	0	0	0	0	0	0	0	0	0
	+ Loans. getLoanAmount	0	0	0	0	0	0	0	0	0	0	0	1

Рисунок 3.4 – Матрица безопасности банковской системы

- Результаты метрик безопасности модели.

Основная цель этого раздела состоит в том, чтобы показать, как вычислить метрики безопасности, определенные этой моделью на основе матрицы безопасности, показанной на рисунке 3.4. Цель и результат каждого показателя поясняются здесь, чтобы определить их влияние на безопасность рассматриваемой здесь банковской системы.

Метрики инкапсуляции данных измеряют соотношение доступных критически важных для безопасности атрибутов и методов в данном классе относительно общего числа критически важных для безопасности атрибутов и методов в этом классе с учетом их значений безопасности. Это означает, что при вычислении этих метрик учитывался уровень безопасности каждого атрибута и метода. Эти значения можно легко получить из матрицы безопасности, показанной на рисунке 3.4. Поэтому ASCAM вычисляется путем подсчета числа доступных (неприватных) критических для безопасности атрибутов, умноженных на каждое из их значений безопасности. В проекте три из десяти атрибутов, определенных на схеме классов UML банковской системы, существует восемь критически важных для безопасности атрибутов. Из этих восьми критически важных для безопасности атрибутов есть два атрибута, которые доступны извне их классов (т.е. Customers.dobи Customers.nationalID). Эта информация показана в матрице безопасности, так как эти два атрибута имеют символ (+) перед ними, который является модификатором доступа для общих атрибутов. Значение безопасности, связанное с Customers.dob равно 2 и с Customers.nationalID - 3. Поэтому общее значение безопасности доступных критически важных для безопасности атрибутов в данном случае равно 5. В полной безопасности, стоимость возможной доступности безопасности важных атрибутов рассчитывается путем умножения общего числа защищенности критически важных атрибутов

в конструкции (т.е. 8) максимальное значение критичности в модели безопасности, используемые в контексте (т.е. 4), которая составляет 32. Следовательно, ASCAM вычисляется путем деления 5 на 32, что составляет 0.16.

Что касается ASCMM который оценивает доступность для обеспечения безопасности особо важных методов в определенной конструкции, вычисляется путем подсчета количества доступных (неприватных) критических для безопасности методов, умноженных на их значение безопасности для каждого из них в общее число возможных доступности критических для безопасности методов в том же проекте, умноженное на значение максимальной безопасности, определенное в модели. Значение безопасности определенного метода совпадает со значением безопасности атрибута, с которым он взаимодействует, которое показано в матрице безопасности. Если существует метод, который взаимодействует с более чем одним атрибутом различных уровней безопасности, то уровень безопасности этого метода совпадает с самым высоким атрибутом, с которым он взаимодействует. Матрица безопасности на рисунке 3.4 показывает, что существует одиннадцать критически важных для безопасности методов, восемь из которых являются доступными критическими для безопасности методами. Из этих восьми методов существует два метода, значение безопасности которых равно 1:Cheques.getChequeNo и Loans.getLoanAmount. Есть один метод, его значение безопасности 2 (т.е. Cheques.getChequeAmount). Один метод,у которого значение безопасности равно 3;Customers.getNationalID. Естьпятьметодов,у которых их значения безопасности равно 4;Customers.getPortalPassword, два метода вклассе Кредитные карты CreditCards.getCardPassword и CreditCards.setCardPassword и два метода в классе Дебетовые карты DebitCards.setCardPassword и DebitCards.verifyCredentials. Общее возможное значение безопасности доступных критических методов безопасности подсчитывается путем умножения общего числа критических методов безопасности (т.е. 11) на максимальное значение критичности в модели безопасности, используемой в контексте (т.е. 4), которое составляет 44. Следовательно, ASCMM вычисляется путем деления 27 на 44, что составляет 0,61.

В терминах метрики, которая измеряет связность критических для безопасности методов в данной конструкции (т.е. CSCMM), она вычисляет взаимодействия методов в данной программе с критическими для безопасности атрибутами в той же программе с учетом значений безопасности этих атрибутов. Основной целью этого показателя является низким значением связности, как это возможно, чтобы соответствовать требованиям конструктивный принцип наименьших привилегий. Из структуры, используемой в этом контексте, существует 11 критически важных для безопасности методов, взаимодействующих с различными критически важными для безопасности атрибутами различных уровней безопасности. Критичность взаимодействий вычисляется путем суммирования числа

взаимодействий в каждом классе, а затем умножения каждого взаимодействия на его значение безопасности. Этот результат делится на число методов в одном классе, умноженное на число критических для безопасности атрибутов, и на максимальное значение безопасности в модели. В случае класса Customers, критичность взаимодействия по безопасности важных атрибутов 9.Это извлечено из матрицы безопасности на рисунке 3.4, так как критичность взаимодействий от клиентов Customers.getDOB - это 2, критичность взаимодействий от клиентов Customers.getNationalID равно 3, и критичность взаимодействий от клиентов Customers.getPortalPassword - 4. Кроме того, общее число возможных взаимодействий внутри класса клиентов своей безопасности важных атрибутов 48. Тот же процесс должен быть сделан для всех классов f программа, то сумма делится на общее количество классов в этой программе. Следовательно, связанность класса кредитных карт составляет 0,33, класса дебетовых карт является 0.6875, класс чеков 0.1875, и класса кредиты является 0,125. Наконец, CSCMM вычисляется путем деления суммы сплоченность всех классов (т. е. 1.5175) число классов (т. е. 5), что дает 0.3035 и это результат CSCMM для конструкции показанной здесь.

Что касается метрики безопасности сцепления, она направлена на сокращение числа ассоциаций классов с другими критически важными для безопасности внутри данной программы. Чем меньше таких ассоциаций, тем более безопасна программа с точки зрения принципа наименьших привилегий. Эта метрика вычисляется путем подсчета количества внешних классов, которые взаимодействуют с критически важными для безопасности атрибутами, и каждое взаимодействие умножается на уровень безопасности этих атрибутов. Уровень безопасности каждого взаимодействия определяется уровнем безопасности атрибута, с которым взаимодействует класс. Затем это число делится на общее число возможных ассоциаций с атрибутами безопасности. На рисунке 3.4 представлены пять критических для безопасности классов, которые имеют десять критически важных для безопасности атрибутов различных уровней безопасности. Число классов, взаимодействующих с критически важными для безопасности атрибутов Customers.dob - это два, а уровень безопасности - два. Следовательно, критичность этих взаимодействий, когда эти два числа умножаются друг на друга - четыре. Число классов, взаимодействующих с критически важными для безопасности атрибутомCustomers.nationalID - четыре, а уровень безопасности - три. Таким образом, критичность этих взаимодействий равна двенадцати. Число классов, взаимодействующих с критическим для безопасности атрибутомCustomers.portalPassword - три, и уровень безопасности - четыре. Таким образом, критичность этих взаимодействий равна двенадцати. Наконец, критичность общего числа связей с классами, критичными для безопасности, равна 28. Критичность возможного числа классов, которые могут взаимодействовать с критичными для безопасности классами в этой конструкции, составляет 160 (число классов меньше на один, умноженное на число критических для безопасности атрибутов, а затем

умноженное на максимальный уровень безопасности в этом случае). Поэтому CSCCM вычисляется путем деления 28 на 160, что равно 0,175.

Метрика размера проекта предназначена для измерения доли критических для безопасности классов в данной программе. Более низкое значение этой метрики желательно для удовлетворения требований принципа проектирования безопасности «уменьшения размера поверхности атаки». Чтобы вычислить эту метрику, каждый критический для безопасности класс умножается на его уровень безопасности, и сумма этого делится на общее число классов, умноженное на максимальное значение безопасности в определенной конструкции. Значение безопасности определенного класса совпадает с наибольшим значением безопасности включенных в него атрибутов. В схеме, показанной на рисунке 3.4, существует пять классов, все из которых имеют критические для безопасности атрибуты. Есть три класса, у которых их значение безопасности равно 4; клиенты, Кредитные карты и дебетовые карты. Существует один критический для обеспечения класс, стоимость обеспечения которого равна 2 (т.е. чеки), и один класс, стоимость обеспечения которого равна 1, т. е. кредиты. Сумма этих значений безопасности 15. Общие значения безопасности всех классов в определенном проекте вычисляются путем умножения числа классов (5 в этом контексте) на максимальное значение безопасности в данном проекте (4 в этом контексте). Следовательно, SCDSM вычисляется путем деления 15 на 20, что составляет 0,75.

3.6 Результат

На рисунке 3.5 показаны результаты пяти показателей безопасности, определенных и изученных в данной главе. В этой части рассматриваются полученные результаты, чтобы показать, какие из них способствуют более безопасной конструкции. Как показано в предыдущем разделе, результаты этих метрик различаются, и это влияет на безопасность программы со ссылкой на соответствующий принцип проектирования безопасности. Например, результат мошенничества является самым низким среди результатов пяти показателей, определенных этой моделью. Поскольку ASCAM измеряет поток информации относительно доступности критически важных для безопасности атрибутов, чтобы удовлетворить требованиям принципа проектирования безопасности «уменьшения размера поверхности атаки», то этот результат указывает, что это-лучший показатель, который делает программу безопасной в этом отношении.

С другой стороны, SCDSM измеряет информационный поток критически важных для безопасности данных в данном объектно-ориентированном проекте с учетом размера критических для безопасности классов в этом проекте. Поэтому желательно иметь как можно меньше критически важных для безопасности классов, чтобы удовлетворить принцип проектирования безопасности «уменьшения размера поверхности атаки». Тем

не менее, результат SCDSM в примере, показанном здесь, является самым высоким среди всех пяти показателей, и, следовательно, рассматриваемая программа является наименее безопасной в этом отношении.

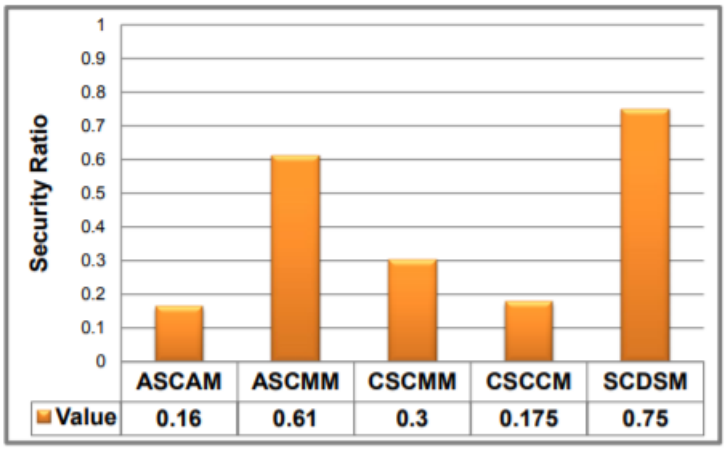

Рисунок 3.5 – График результатов банковской системы

С точки зрения метрик, которые измеряют поток информации в данном проекте, чтобы соответствовать требованиям принципа проектирования безопасности предоставления наименьших привилегий, результаты этих метрик показывают, что CSCCM ниже, чем CSCMM. Это указывает на то, что CSCCM является более безопасным, чем CSCMM с точки зрения наименьшего принципа проектирования безопасности привилегий в этом случае.

4РеализациямногоуровневоймоделиразграничениядоступавбазахданныхMicrosoftSQLServer

Базыданных (БД) всовременном мире широко используются в прикладном программном обеспечении и web-приложениях, предоставляя удобное решение для хранения информации.В большинстве своем сохранность информации и ее неприкосновенность от несанкционированных изменений, находящейся и используемой в данный информационных системах, является одной из главных проблем, так как потеря ее может принести владельцу серьезные проблемы, которые выльются в убытки.

Администраторы, отвечающие за безопасностьБД, занимаются настройкой (изменением), сопровождением и защитой БД. Одной из задач администрирования БД является управление доступом пользователей к данным и другим объектам БД. В ходе управления доступом принимается решение о том, какие действия в системе может выполнять пользователь, а также какие операции разрешено выполнять приложениям, запущенным от имени пользователя. Таким образом, правильное управление доступом базой данных может предотвратить умышленные или неумышленные действия пользователя, которые способны нанести вред хранимой информации [25].

Обеспечение безопасности информации при функционировании любой информационно-управляющей системы (АИУС) является важным вопросом в решении вопросов безопасности. Эффективность работы АУИС может быть значительно снижена при наличии ошибок в процессе проектирования системы защиты информации от несанкционированного доступа. Защита информации от несанкционированного доступа осуществляется для обеспечения целостности данных, исключающее их уничтожение или повреждение; для недопущения несанкционированного изменения данных, представляющего собой умышленные или случайные действия по модификации информации; для недопущения несанкционированного получения данных и для недопущения несанкционированного тиражирования данных.

К современным информационным системам часто предъявляется требование разграничения доступа пользователей не только к объектам БД, но и к отдельным записям, с применением многоуровневых моделей доступа. По этой причине целесообразно реализовывать программный механизм, реализующий многоуровневую модель управления доступом на уровне записей, на стороне систем управления базами данных (СУБД). Большая часть современных реляционных СУБД не имеет в своем составе такого механизма, оставшаяся часть содержит в себе этот механизм в качестве дополнительного инструмента, требующего дополнительной оплаты и имеющего ограниченную функциональность.

Механизм управления доступом в базах данных служит для обеспечения информационной безопасности СБД. Программные методы защиты информации позволяют реализовать большинство функций защиты:

идентификацию пользователей БД, проверку подлинности пользователей, проверку полномочий пользователей на осуществление того или иного типа доступа к защищаемым данным, реагирование на попытку несанкционированного доступа путём прерывания выполнения запроса или отключения терминала, уничтожение остаточной информации в оперативной памяти компьютеров после удовлетворения санкционированного запроса пользователя и др.

Кроме того, они могут поддерживать различные стратегии разграничения доступа пользователей к ресурсам БД: разграничение по спискам, разграничение по матрицам установления полномочий, разграничение по кольцам секретности, мандатную модель доступа. Механизмы защиты (МЗ) применяются с целью организации доступа и допуска к защищенной информации только тех групп пользователей, которые обладают соответствующими полномочиями и прошли авторизацию.

В качестве базовой модели доступа в СУБД Microsoft SQL Server используется ролевая модель. Правила доступа в ней описываются в виде троек. Субъектами доступа (участниками) могут быть учетные записи серверного уровня, пользователи баз данных, серверные роли и роли баз данных. Учетные записи и серверные роли наделяются привилегиями серверного уровня, каждая из которых дает возможность выполнять то или иное действие на сервере: устанавливать соединение с сервером, просматривать список имеющихся баз данных, создавать новые базы данных, изменять свойства существующих баз данных или удалять их, управлять настройками сервера (в т.ч. настройками безопасности) и т.п. Пользователи и роли баз данных наделяются привилегиями доступа к объектам тех баз данных, в которых они зарегистрированы: привилегиями чтения/записи информации в таблицах, запуска хранимых процедур, изменения свойств объектов базы данных, управления безопасностью на уровне базы данных и т.п. Недостатком базовой модели доступа является невозможность разграничения доступа пользователей к отдельным строкам и ячейкам таблиц, а также отсутствие многоуровневого управления доступом. Для устранения данного недостатка требуется расширение имеющейся подсистемы управления доступом в виде механизма, поддерживающего указанные функциональные возможности.

4.1 Меточная политика доступа как основа механизма многоуровневого управления доступом

Компанией Microsoft представлены рекомендации по созданию механизма многоуровневого управления доступом. В документации предлагается настраивать безопасность на уровне строк с использованием представлений и использовать метки безопасности (security labels).

Под меткой безопасности понимается та часть информации, которая описывает «чувствительность» (sensitivity) элемента данных (строки). Строка

маркируется меткой, описывающей уровни доступа, классифицируемые по одному или нескольким признакам. Пользователи (субъекты) имеют разрешения, описанные с теми же метками. Разрешения каждого субъекта могут рассматриваться в качестве собственных меток. Метка субъекта сравнивается с меткой строки для определения возможности доступа к этому строке.

Привязка метки безопасности к пользователю определяет, к каким строкам таблицы он может получить доступ.

Пусть имеется множество строк (таблица 4.1), содержащих метки безопасности в столбце Classification.

Таблица 4.1 – Таблица ролей

ID	Name	Classification
1	Ivan Ivanov	SECRET
2	Peter Petrov	TOP SECRET
3	Michael Sidorov	UNCLASSIFIED

Также в базе данных может храниться информация о пользователях (таблица 4.2):

Таблица 4.2 – Таблица пользователей

User	Clearance
Anna	SECRET
Alex	UNCLASSIFIED (no clearance)

Обапользователя, AnnaиAlex, могутвыполнитьзапроснавыборкувидаSELECT * FROMктаблицеданных (таблица 4.1), норезультатыданногозапросавыглядятнеодинаково. ПользовательAnnaполучаетнаборданных, представленныйвтаблице4.3, пользовательAlex - представленныйвтаблице 4.4.

Таблица4.3 – Пользователь Anna

ID	Name	Classification
1	Ivan Ivanov	SECRET
3	Michael Sidorov	UNCLASSIFIED

Таблица4.4 – Пользователь Alex

ID	Name	Classification
3	Michael Sidorov	UNCLASSIFIED

В метку безопасности могут быть включены уровни доступа по нескольким категориям. Например, в дополнение к классификации секретности (с уровнями SECRET, TOP SECRET и UNCLASSIFIED) может

использоваться такая категория, как членство пользователя в определенной проектной команде. Предположим, имеется группа PROJECT Q. Изменим данные в таблице 4.1 таким образом, чтобы метка для первой строки включала уровень секретности SECRET и задавала разрешение на доступ к этой строке только для участников группы PROJECT Q. Результат этого изменения представлен в таблице 4.5.

Таблица 4.5 – Результат изменения

ID	Name	Classification
1	Ivan Ivanov	SECRET, PROJECT Q
2	Peter Petrov	TOP SECRET
3	Michael Sidorov	UNCLASSIFIED

Теперь изменим права пользователей: включим пользователя Anna в группу PROJECT Q и добавим пользователя Charlie с уровнем безопасности TOP SECRET (таблица 4.6).

Таблица 4.6 – Добавления пользователя Charlie

User	Clearance
Anna	SECRET, PROJECT Q
Alex	UNCLASSIFIED (no clearance)
Charlie	TOP SECRET

Выполняя запрос вида SELECT * FROM , Anna видит результаты, представленные в таблице 4.7, а Charlie - в таблице 4.8. Хотя Charlie имеет уровень доступа TOP SECRET, он не включен в группу PROJECT Q, поэтому он не может видеть строку 1. Anna, однако, имеет все права для того чтобы видеть строку 1. Строка 2, требующая метку TOP SECRET, видна только Charlie.

Таблица 4.7 – Выполнение запроса для пользователя Anna

ID	Name	Classification
1	Ivan Ivanov	SECRET, PROJECT Q
3	Michael Sidorov	UNCLASSIFIED

Таблица 4.8 - Выполнение запроса для пользователяCharlie

ID	Name	Classification
2	Peter Petrov	TOP SECRET
3	Michael Sidorov	UNCLASSIFIED

В некоторых областях применения меточной политики разграничения доступа метки безопасности могут включать в себя несколько классов доступа, относящихся к различным категориям, а число возможных

комбинаций этих классов меток может быть достаточно большим. В этой связи уместно рассмотреть рекомендуемую компанией Microsoft архитектуру механизма многоуровневого контроля доступа, ориентированную на использование различных (задаваемых администратором) категорий и классов доступа при определении меток безопасности.

Для настройки безопасности предлагается использовать следующие служебные таблицы базы данных (рисунок 4.1).

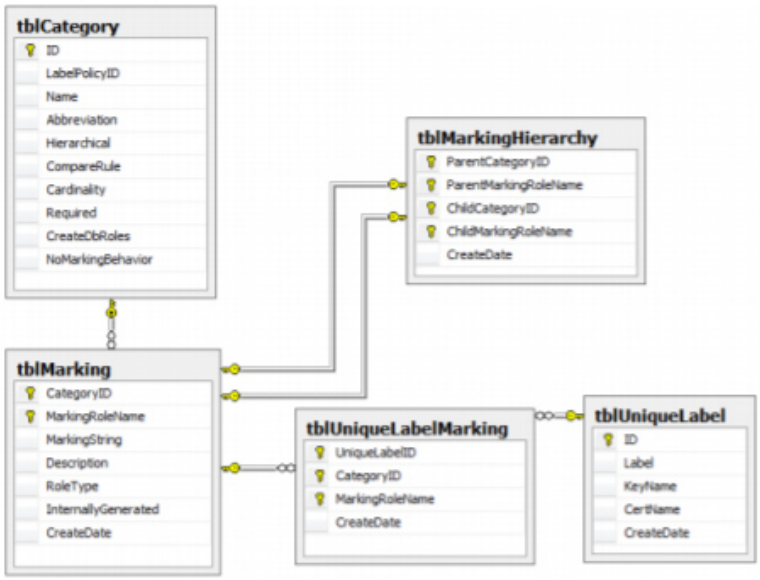

Рисунок 4.1 – Диаграмма БД

В таблице tblCategory предлагается хранить категории разграничения доступа. В качестве таковых категорий может выступать уровень защищенности информации, группа пользователей и т.п.

Для каждой категории существует определенный набор возможных маркировок, которые предлагается хранить в таблице tblMarking.

Если категория имеет иерархическую структуру, то отношения «родитель-потомок» устанавливаются с соответствующими записями в таблице tblMarkingHierarchy.

Таблица tblUniqueLabel используется для хранения меток безопасности. Эта таблицу рекомендуется заполнять «по требованию»: при добавлении записей с некоторой меткой безопасности сначала производить поиск этой метки в таблице tblUniqueLabel с помощью специальной хранимой процедуры; при неудачном завершении поиска метка добавляется в таблицу.

В таблице tblUniqueLabelMarking отдельные значения меток связываются с экземплярами меток безопасности.

Помимо таблиц, рекомендуется включить в архитектуру служебной БД некоторые программные объекты, поддерживающие контроль безопасности на уровне отдельных записей в автоматическом режиме.

Прежде всего, к таковым объектам относится представление vwVisibleLabels, которое позволяет получить набор меток, доступных для текущего пользователя.

Листинг представления имеет следующий вид:

```
CREATE VIEW vwVisibleLabels
AS
SELECT ID, Label
FROM tblUniqueLabel WITH (NOLOCK)
WHERE ID IN --Classification
(SELECT UniqueLabelID FROM tblUniqueLabelMarking WITH (NOLOCK)
WHERE CategoryID = 1 AND IS_MEMBER(MarkingRoleName) = 1)
AND –Compartments
1 = ALL(SELECT IS_MEMBER(MarkingRoleName) FROM tblUniqueLabelMarking
WHERE CategoryID = 2 AND UniqueLabelID = tblUniqueLabel.ID)
GO
```

Кроме того, требуется обеспечить опосредованный доступ на чтение и запись к пользовательской таблице, чтобы в процессе чтения и записи проводилось сравнение меток безопасности пользователя с метками обрабатываемых строк. Главным объектом здесь является представление для пользовательской таблицы, к которому пользователь должен выполнять запросы на чтение или запись. Представление же построено на запросе к пользовательской таблице, и в этом запросе происходит сравнение меток безопасности, благодаря чему меточная безопасность на уровне строк таблицы контролируется в прозрачном режиме.

Недостатком рассмотренной модели является избыточность (составные внешние ключи в таблицах tblMarkingHierarchy, tblUniqueLabelMarking), дублирование информации о пользовательских ролях БД (кроме того, что эти роли заранее создаются с помощью встроенных средств SQL Server, информация о них должна быть дополнительно представлена в таблицах tblMarking и tblMarkingHierarchy, что может существенно усложнить настройку механизма управления доступом). Рассмотренное выше представление vwVisibleLabels позволяет определять доступные метки для текущего пользователя только по двум категориям, хотя наличие таблицы tblCategory в архитектуре служебной БД указывает на то, что число категорий не ограничено.

Таким образом, после работы с предложенной схемой построения служебных таблицы была получена оптимизированная схема служебных таблиц для организации меточной модели разграничения доступа (рисунок 4.2).

Данная оптимизация была получена путем устранения составных внешних ключей в таблицах tblMarkingHierarchy, tblUniqueLabelMarking, а также за счет устранения дублирования информации о пользовательских ролях БД в таблицах схемы tblMarking и tblMarkingHierarchy. Как можно заметить, была убрана таблица tblMarkingHierarchy, хранящая информацию об иерархии меток ролей, т.к. вся иерархия хранится в системных таблицах и может быть получена при правильно составленном запросе с использованием служебных представлений select * from sys.Database_Role_Members и select * from Sys.Database_Principals.

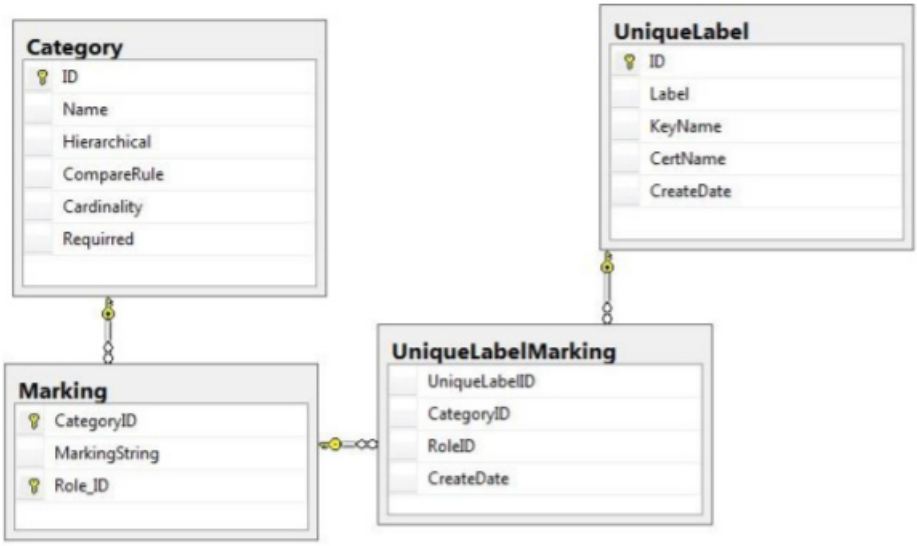

Рисунок 4.2 – Оптимизированная схема служебных таблиц

Для автоматизации процесса настройки разграничения доступа к пользовательским таблицам был написан ряд программных объектов.

Для автоматизации процесса заполнения служебных таблиц была создана процедура заполнения таблицы UniqueLabelMarking, в которой в качестве параметров передаются идентификатор метки, уровень безопасности и имя роли:

```
create procedure fillingULM (@LabelNumber int, @CategoryName varchar(30), @RoleName varchar(30))
 as
begin
declare @CategoryNumber int
declare @RoleNumber int
 set @CategoryNumber = (select Category.ID from Category where Category.Name = @CategoryName)
```

set @RoleNumber = (select Marking.Role_ID from Marking,sysusers where Marking.CategoryID=@CategoryNumber and sysusers.name = @RoleName and sysusers.uid=Role_ID)

insert into UniqueLabelMarking values (@LabelNumber,@CategoryNumber,@RoleNumber," ")

end

Заполненные служебные таблицы имеют следующий вид и представлены на рисунках 4.3-4.6.

	ID	Name	Hierarchical	CompareRule	Cardinality	Required
1	1	Classification	1	Any	One	1
2	2	Compartment	0	All	Many	0
3	3	Nationality	0	Any	Many	0
4	4	Need-to-Know	0	Any	Many	0

Рисунок 4.3 – Служебная таблица Category

	CategoryID	MarkingString	Role_ID	Mark
1	1	SECRET	5	0
2	1	CONFIDENTIONAL	6	1
3	1	UNCLASSIFIED	9	2
4	1	TOPSECRET	10	3

Рисунок 4.4 – Служебная таблица Marking

	ID	Label	KeyName	CertName
1	1	L1	K1	Cert1
2	2	L2	K2	Cert2
3	3	L3	K3	Cert3
4	4	L4	K4	Cert4

Рисунок 4.5 – Служебная таблица UniquelabelMarking

	UniqueLabelID	CategoryID	RoleID
1	1	1	10
2	2	1	5
3	3	1	6
4	4	1	9
5	4	1	5
6	3	1	6

Рисунок 4.6 – Служебная таблицаUniquelabel

Кроме того, на основе рекомендаций было написано и модифицировано представление VisibleLabels, которое обеспечивает получение набора меток,

доступных текущему пользователю. Главное отличие данного представления от оригинального заключается в использовании в качестве параметра определения принадлежности пользователя к роли специальной функции getRoleName (RoleID), где RoleID - идентификатор роли. Функция getRoleName (RoleID) имеетследующуюструктуру:

```
create function getRoleName (@ID int) returns varchar(max)
as
 begin
return (select name from sysusers,Marking where uid=@ID and Marking.Role_ID=@ID)
end
```

Представлениеимеетвид:

```
create view VisibleLabels
as
selectID, Label
fromUniqueLabelwith (nolock)
whereIDin --Classification
(selectUniqueLabelIDfromUniqueLabelMarkingwith (nolock)
whereCategoryID = 1 andIS_MEMBER(dbo.getRoleName(RoleID)) = 1)
and --Compartments
1 = ALL(selectIS_MEMBER(dbo.getRoleName(RoleID)) FROMUniqueLabelMarking
whereCategoryID = 2 andUniqueLabelID = UniqueLabel.ID)
```

Результат работы представления, когда к нему обращаются пользователи с разными уровнями доступа, выглядит следующим образом:

Для пользователя UserS с уровнем доступа «Секретно».

	ID	Label
1	2	L2
2	3	L3
3	4	L4

Рисунок 4.7 – Результат представления VisibleLabelsдля пользователя UserS

Для пользователя UserC с уровнем доступа «Конфиденциально»:

	ID	Label
1	3	L3
2	4	L4

Рисунок 4.8 – Результат представления VisibleLabelsдля пользователя UserC

4.2 Bank Account Management System

Bank Account Management System или система управления банковскими счетами – это приложение, разработанное на языке программирования Java в среде NetBeansIDE8.2, для ведения счета лица в банке. В этом проекте я попыталась показать работу системы управления банковских счетов с использованием модели безопасности Белл-ЛаПадула. Разработать проект для решения финансовых приложений клиента в банковской среде, чтобы выполнять потребности конечного пользователя, предоставляя различные способы выполнения банковских задач.

На основе разработанной системы проводилась оценка многоуровневой безопасности в главе 3 с помощью общей модели.Разработанный программный продукт на языке Java соединен с SQLiteManager, код программы изложен в приложении А.

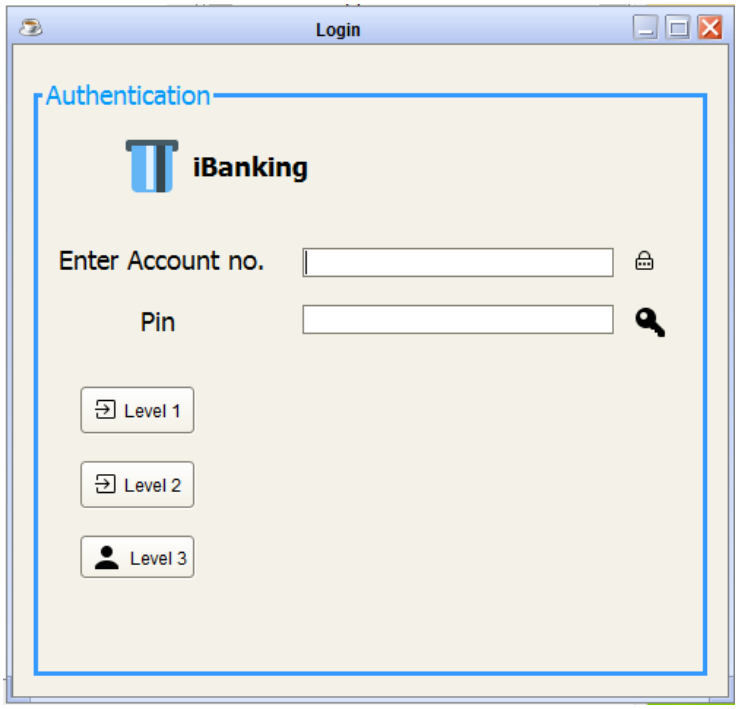

Рисунок 4.9 –Главная страница программного продукта Bank Account Management System

На главном окне программы (рисунок 4.9) расположены уровни доступа менеджеров банка с вводом логина и пароля. Базу данных, точнее таблицы,

использованные в данной программе можно посмотреть на рисунке 4.10. Код программы с разграничением доступа представлен в приложении В.

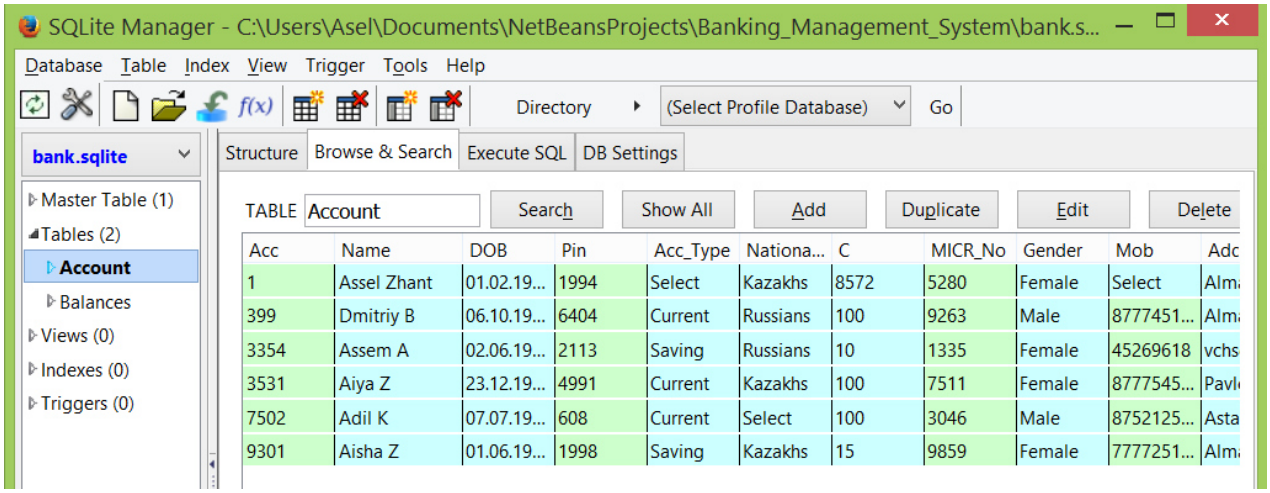

Рисунок 4.10 – Отображение таблиц БД в SQLiteManager

При неправильном вводе данных или же при попытке доступа к данным, которым пользователь не имеет доступа, выводится сообщение об ошибке (рисунок 4.11).

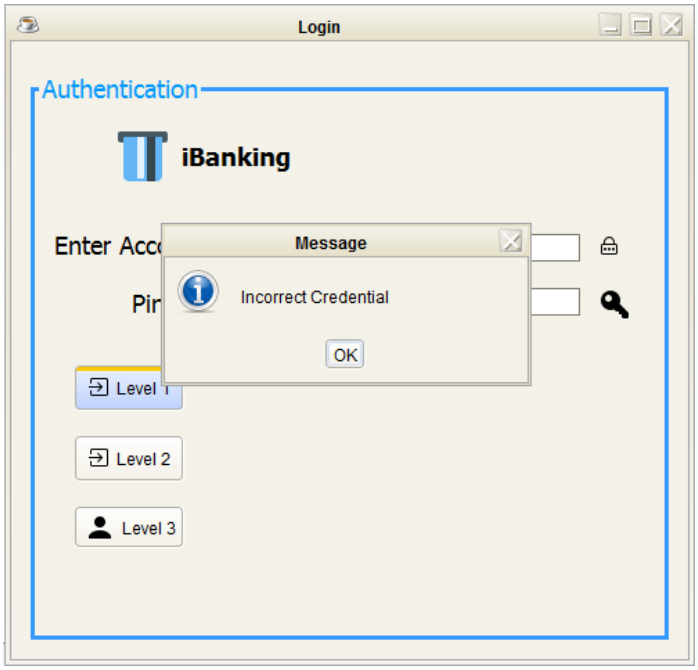

Рисунок 4.11 – Вывод сообщения при несанкционированном доступе

Окно пользователя с самым низким уровнем доступа представлено на рисунке 4.12. Данный пользователь имеет право просматривать данные клиента банка (за исключением его PIN-кода), если он имеется в базе данных.

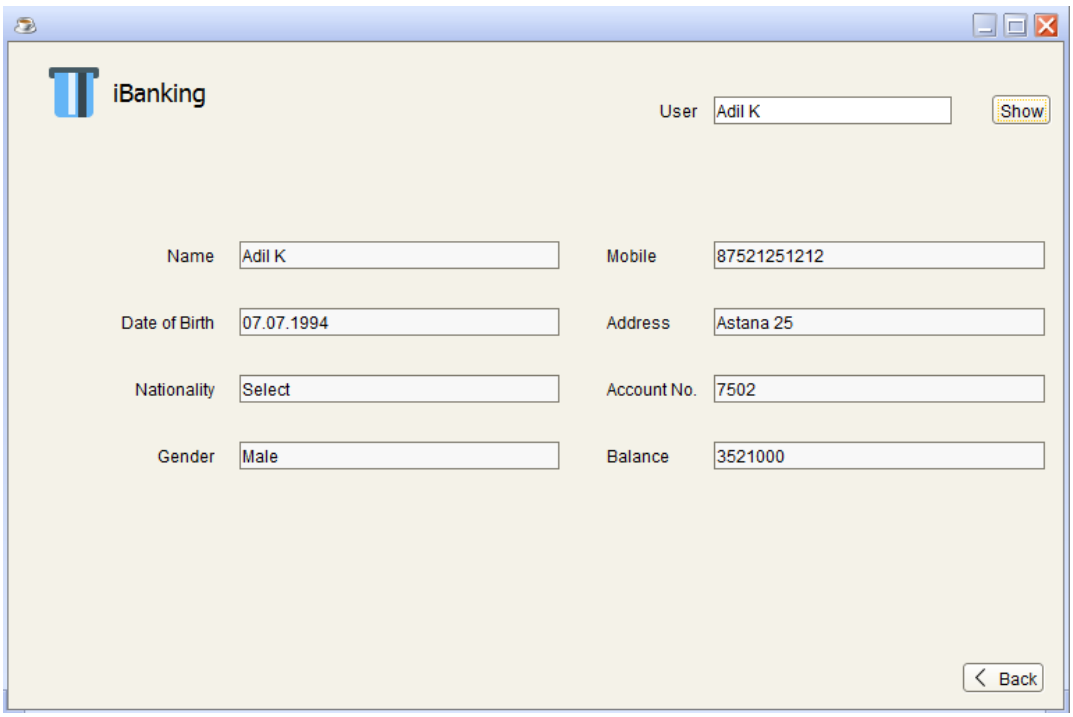

Рисунок 4.12 – Окно с самым низком уровнем доступа

Перед доступом на секретный уровень (рисунок 4.14) загружается окно «Загрузки» (рисунок 4.13).

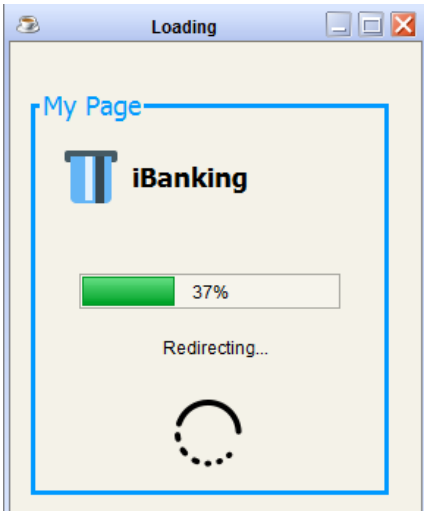

Рисунок 4.13 – Окно загрузки

В данном окне имеются следующие вкладки: просмотр профайла клиента, «Депозит», «Перевод» денежных средств между клиента банка, «Изъятие» денег клиентом банка, таблица всех клиентов банка, таблица счетов с балансом, изменение PIN-кода.

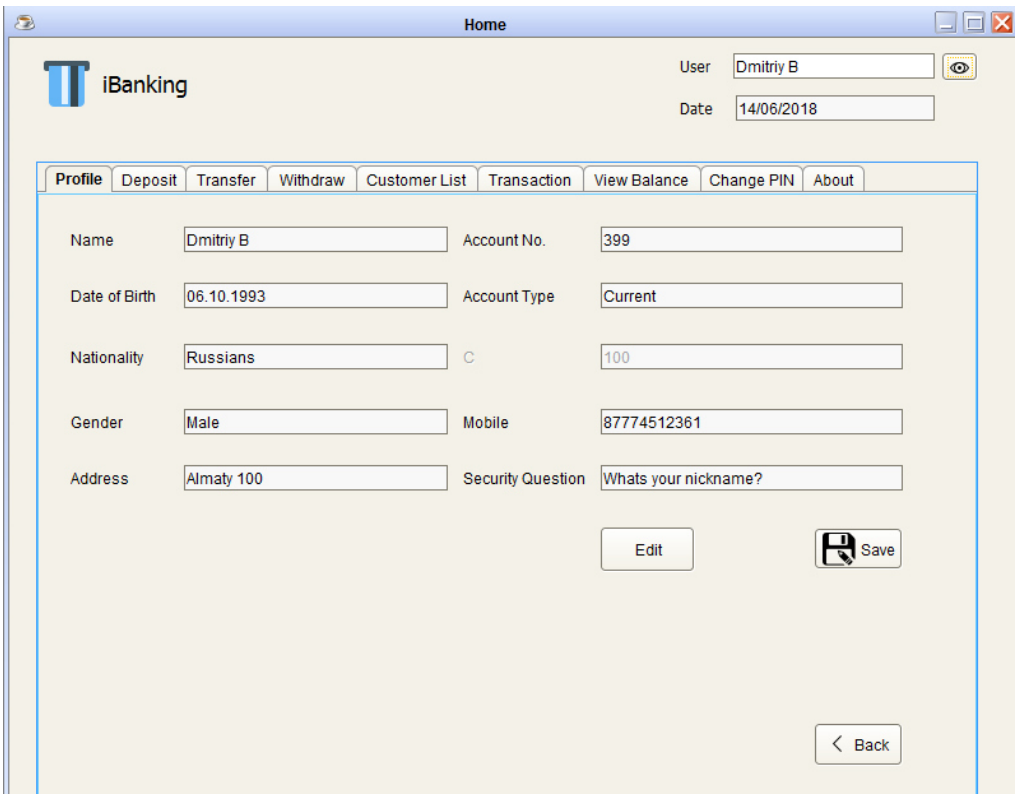

Рисунок 4.14 – Окно с секретным уровнем доступа

Окно «Депозит» программного продукта Bank Account Management System представлено на рисунке 4.15.

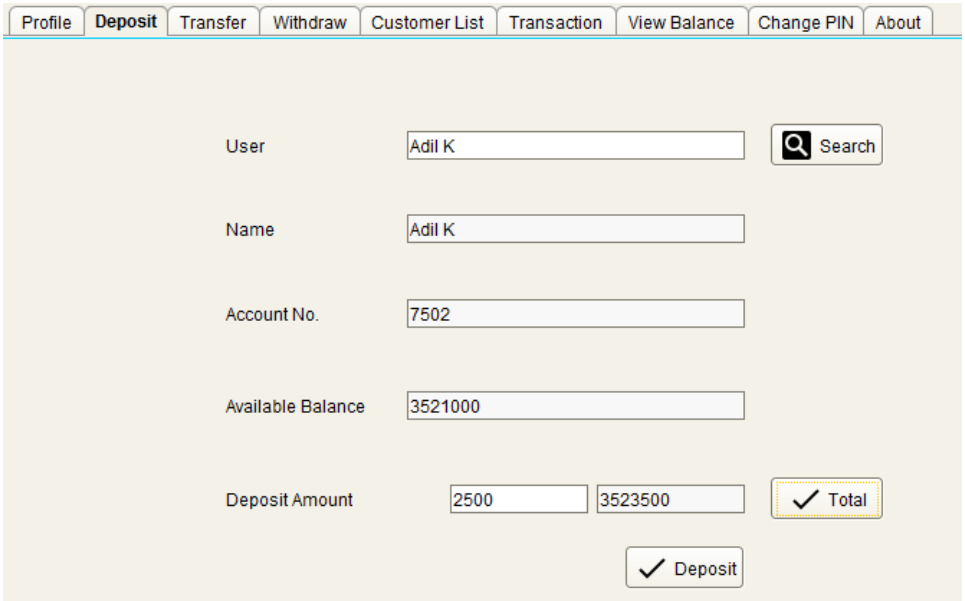

Рисунок 4.15 – Окно работы с вкладкой «Депозит»

Окно «Депозит» программного продукта Bank Account Management System представлено на рисунке 4.16.

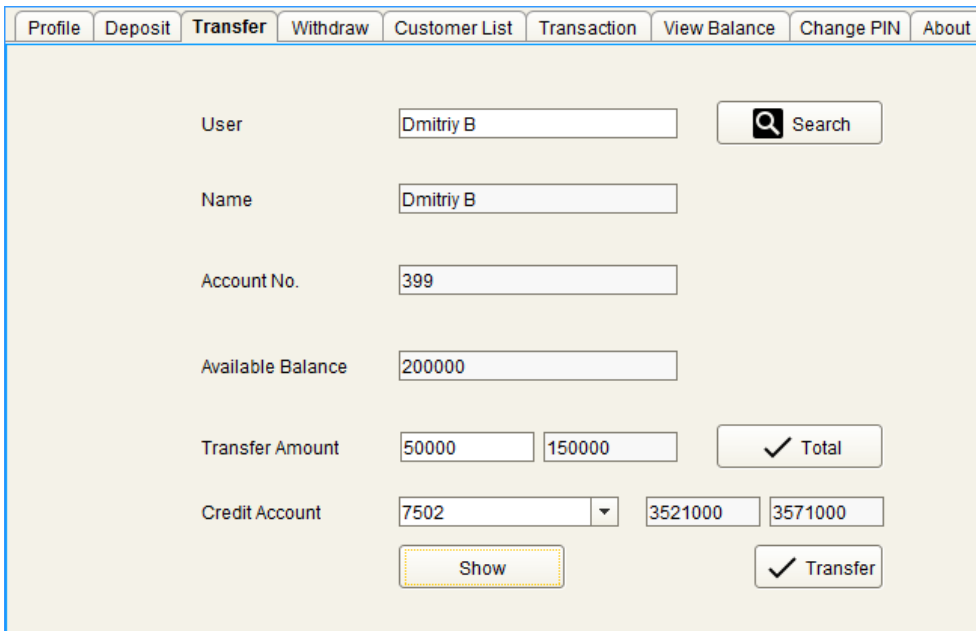

Рисунок 4.16 – Окно работы с вкладкой «Перевод»

Сообщение об успешном завершении операции с счетом клиента в банке представлено на рисунке 4.17

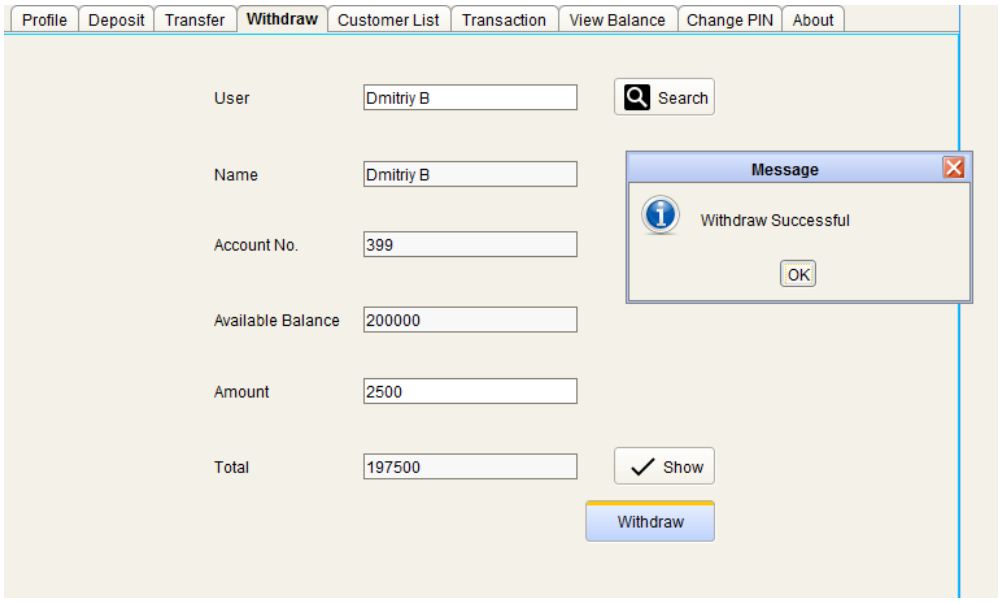

Рисунок 4.17 – Сообщение об успешном завершении операции с счетом клиента в банке

Для проверки об корректном изъятии денег и сохранении изменений в БД со счета клиентом DmitriyB приводится на рисунке 4.18

Structure | Browse & Search | Execute SQL | DB Settings

TABLE Balances | Search | Show All | Add | Duplicate | Edit

rowid	Name	Acc	MICR_No	Balance
2	Assel Zhant	8572	5280	1100
3	Aisha Z	9301	9859	2500
4	Assem A	3354	1335	2500
5	Dmitriy B	399	9263	197500

Рисунок 4.18 – Проверка об изменении данных клиента в БД

Profile | Deposit | Transfer | Withdraw | Customer List | Transaction | View Balance | Change PIN | About

Acc	Name	DOB	Acc_Type	Gender	Mob
1	Assel Zhant	01.02.1994	Select	Female	Select
399	Dmitriy B	06.10.1993	Current	Male	87774512361
3354	Assem A	02.06.1963	Saving	Female	45269618
3531	Aiya Z	23.12.1990	Current	Female	87775452121
7502	Adil K	07.07.1994	Current	Male	87521251212

Рисунок 4.19 – Окно просмотра списка клиентов банка для менеджера

Пользователь со сверхсекретным уровнем безопасности (рисунок 4.20) имеет права на добавлении клиентов и сотрудников банка с присвоением им определенных уровней доступа (рисунок 4.20).

Create Account

iBanking

New Account

Account no. 1021 | Name

MICR no. 2649 | Date of Birth

Pin 6805 | Nationality Select

Account Type Select | C

Gender Male Female | Mobile

Address | Security Q. Whats your nickname?

Amount | Answer

Create | Back | Clear

Thank You for Banking with Us.

Рисунок 4.20 – Окно пользователя со сверхсекретным уровнем безопасности

Заключение

Несмотря на неудачи и разочарования, которые преследовали разработки продуктов MLS за прошлую четверть века, конечные пользователи все еще призывают к возможностям MLS. Это связано с тем, что проблема остается: оборонное сообщество должно обмениваться информацией на нескольких уровнях безопасности. Большая часть сообщества решает эту проблему, работая над многоуровневыми данными в среде с высоким уровнем системы и занимаясь проблемами понижения рейтинга по частям. В то время как это решает проблему в некоторых ситуациях, это не практично другие, как «датчик-стрелок» приложений.

Классические стратегии, предназначенные для получения продуктов MLS, потерпели неудачу несколькими способами. Во-первых, правительство провалило продвижение оценок продуктов, когда поставщики обнаружили, что возможности MLS не значительно увеличивают продажи продуктов. Концепция развертывания доказуемо безопасной системы потерпела неудачу дважды: во-первых, когда поставщики обнаружили, насколько дорогостоящими и неопределенными могут быть оценки, особенно на самых высоких уровнях, и во-вторых, когда эксперты по безопасности обнаружили, насколько неразрешимой может быть проблема скрытого канала. Наконец, несколько продуктов MLS, которые пробивались на рынок, томились, когда конечные пользователи поняли, насколько узко продукты решали свои проблемы безопасности и совместного использования. Основные успехи в MLS сегодня основаны на защите и доверия серверным продуктам

В данной работе была предоставлена общая модель для оценки многоуровневой безопасности объектно-ориентированных программ.

Список литературы

1 Anderson, J.P. (1972). Computer Security Technology Planning Study Volume II, ESD-TR-73-51, Vol. II. Bedford, MA: Electronic Systems Division, Air Force Systems Command, Hanscom Field.

2 Bell, D.D. and L.J. La Padula (1974). Secure Computer System: Unified Exposition and Multics Interpretation, ESD-TR-75-306. Bedford, MA: ESD/AFSC, Hanscom AFB.

3 Boehm, B.W. 1981, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall.

4 Brooks, F.P. (1975). *The Mythical Man-Month*. Reading, MA: Addison-Wesley.

5 Byers, S (2004). Information leakage caused by hidden data in published documents. *IEEE Security and Privacy 2* (2) 23-27.

6 Cohen, F.C. (1994) *Short Course on Computer Viruses*, Second Edition. New York: John Wiley & Sons, pp. 35-36.

7 Cohen, F.C. (1990) Computer Viruses. *Computer Security Encyclopedia*.

8 Common Criteria Project Sponsoring Organizations (1999). Common criteria for information technology security evaluation, version 2.1.

9 Department of Defense (1997). DOD Information Technology Security Certification and Accreditation, DOD Instruction 5200.40. Washington, DC: Department of Defense.

10 Department of Defense (1985a). *Trusted Computer System Evaluation Criteria* (Orange Book), DOD 5200.28-STD. Washington, DC: Department of Defense.

11 Department of Defense (1985b). “Specification Practices,” MIL-STD 490A, 4 June 1985. Washington, DC: Department of Defense.

12 Federation of American Scientists (1997). Imagery Intelligence: FAS Space Policy Project – Desert Star.

13 Fine, T. (1996). Defining noninterference in the temporal logic of actions. *Proc. 1996 IEEE Conference on Security and Privacy*. pp. 12-21.

14 Fraim, L.J. SCOMP: a solution to the multilevel security problem. *IEEE Computer 16* (7) 26-34.

15 Haigh, J.T., and Young, W.D. (1987). Extending the noninterference version of MLS for SAT. *IEEE Transactions on Software Engineering SE-13* (2).

16 Hoffman, L.J. (1973). IBM’s Resource Security System (RSS). In L.J. Hoffman (ed.), *Security and Privacy in Computer Systems*(pp. 379-401). Los Angeles: Melville Publishing Company.

17 Kang, M.H., Moskowitz, I.S., and Lee, D.C. (1996). A network pump. *IEEE Transactions on Software Engineering 22* (5) 329-338.

18 Karger, P.A. and R.R. Schell (1974). MULTICS Security Evaluation, Volume II: Vulnerability Analysis, ESD-TR-74-193, Vol. II. Bedford, MA: Electronic Systems Division, Air Force Systems Command, Hanscom Field.

19 Kemmerer, R.A. (2002). A practical approach to identifying storage and timing channels: twenty years later. *Proceedings of the 18th Annual Computer Applications Security Conference*.
20 Kemmerer, R.A. (1983). Shared resource matrix methodology: an approach to identifying storage and timing channels. *ACM Transactions on Computer Systems 1* 109-118.
21 Lampson, B. (1973). A note on the confinement problem. *Communications of the ACM 16* 10, pp 613-615.
22 LaPadula, L.J., LeMoine, J.E., Vukelich, D.F. and Woodward, J.P.L. (1990). DNSIX Detailed Design Specifications, Version 2. Bedford, MA: MITRE Corporation.
23 Nibaldi, G.H., (1979). Proposed Technical Evaluation Criteria for Trusted Computer Systems, M79-225. Bedford, MA: The Mitre Corporation.
24 Orlowski, A. (2004). MS Trusted Computing back to drawing board. *The Register*, May 6, 2004.
25 Исаева М.Ф., Глухарев М.Л., Ветлугин К.А. Реализация многоуровневой модели разграничения доступа в базах данных под управлением системы управления базами данных Microsoft SQL Server // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 9, №3 (2017) http://naukovedenie.ru/PDF/36TVN317.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ
26 Proctor, N.E., and Neumann, P.G. (1992). Architectural implications of covert channels. *Proceedings of the Fifteenth National Computer Security Conference* pp. 28-43.
27 St. Johns, M. (1988). Draft Revised IP Security Option, RFC 1038.
28 Saydjari, O.S. (2004). Multilevel security: reprise. *IEEE Security and Privacy 2* (no. 5). pp. 64-67.
29 Saydjari, O.S. (2002). LOCK: an historical perspective. *Proceedings of the 2002 Annual Computer Security Applications Conference* pp.
30 Saydjari, O.S., Beckman, J.K., Jeffrey R. Leaman, J.R. (1989). LOCK Trek: navigating uncharted space. *Proceedings of the 1989 IEEE Symposium on Security and Privacy* 167-175.
31 Smith, R.E. (2005). Observations on multi-level security.
32 Smith, R.E. (2001). Cost profile of a highly assured, secure operating system. *ACM Transactions on Information System Security 4* pp. 72-101.
33 Smith, R.E. (2000). Trends in government endorsed security product evaluations, *Proceedings of the 23rd National Information Systems Security Conference*.
34 Smith, R.E. (1994). Constructing a high assurance mail guard. *Proceedings of the 17th National Computer Security Conference* 247-253.
35 Ware, W.H. (1970) Security Controls for Computer Systems (U): Report of Defense Science Board Task Force on Computer Security. Santa Monica, CA: The RAND Corporation.

Приложение А

(объязательное)

Текст программы

```
import java.sql.*;
import javax.swing.JOptionPane;
public class javaconnect {
   Connection conn=null;

   public static Connection ConnecrDB (){
      try {
         Class.forName("org.sqlite.JDBC");
         Connection
conn=DriverManager.getConnection("jdbc:sqlite:C:\\Users\\Asel\\Documents\\Net
BeansProjects\\Banking_Management_System\\bank.sqlite");
      return conn;
      } catch(Exception e){
         JOptionPane.showMessageDialog(null, e);
      }
  return null;
}
}
```

Приложение В

(объязательное)

Текст программы

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String sql="select * from Account where Acc=? and Pin=? and C=10";
    try{
      pst=conn.prepareStatement(sql);
      pst.setString(1, jTextField1.getText());
      pst.setString(2, jTextField2.getText());
      rs=pst.executeQuery();
      if(rs.next()){
         setVisible(false);
         Account ob=new Account();
        /*ob.setUpLoading();*/
         ob.setVisible(true);
         rs.close();
         pst.close();
      }
            else{
         JOptionPane.showMessageDialog(null, "Incorrect Credential");
      }
    }catch(Exception e){
       JOptionPane.showMessageDialog(null, e);
    }finally{
       try{
        rs.close();
        pst.close();
       }catch(Exception e){

       }
    }

  }

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String sql="select * from Account where Acc=? and Pin=? and C=15";
    try{
      pst=conn.prepareStatement(sql);
```

```
        pst.setString(1, jTextField1.getText());
        pst.setString(2, jTextField2.getText());
        rs=pst.executeQuery();
        if(rs.next()){
            setVisible(false);
            Loading ob=new Loading();
            ob.setUpLoading();
            ob.setVisible(true);
            rs.close();
            pst.close();
        }
                else{
            JOptionPane.showMessageDialog(null, "Incorrect Credential");
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }finally{
        try{
         rs.close();
         pst.close();
        }catch(Exception e){

        }
    }



  }

  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     String sql="select * from Account where Acc=? and Pin=? and C<>15 and
C<>10 and c<>100";
    try{
        pst=conn.prepareStatement(sql);
        pst.setString(1, jTextField1.getText());
        pst.setString(2, jTextField2.getText());
        rs=pst.executeQuery();
        if(rs.next()){
            setVisible(false);
            Page ob=new Page();
```

```
            ob.setVisible(true);
          rs.close();
          pst.close();
        }
                else{
          JOptionPane.showMessageDialog(null, "Incorrect Credential");
        }
    }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }finally{
        try{
         rs.close();
         pst.close();
        }catch(Exception e){

        }
    }
}
```