**ак семинар**

модели безопасности в ОС

Безопасность является важным аспектом любого проекта, однако, в нашей образовательной программе ей уделено совсем немного, хотя разработчики любой сферы должны иметь понимание, как обезопасить свои системы и данные пользователей. в этом докладе я хочу рассказать о таких моделях безопасности, как Bell-LaPadula, Biba, Clark-Wilson, RBAC, MAC, DAC и ABAC, их сходствах и отличиях, о том, какие модели применяются в реальных ОС (DAC, MAC, RBAC, ABAC, zero Trust, Sandboxing), об уязвимостях в соответствующих моделях безопасности и о том, какие модели могут быть реализованы в будущем(AI-driven Access Control, Self-healing).

Важный ВОРНИНГ  
каждый раз когда я упоминаю использование модели в определенной компании я имею в виду что они реализуются не в чистом виде, но их принципы лежат в основе модели безопасности компании

**Дак**

Дискреционная модель доступа

Суть в том, что создаются списки на использование определенных объектов с указанием их пользователей

Права определяет владелец объекта, можно быстро выдавать и отзывать доступ, все решают люди

Работает как базовая rwx с опд

Считаются самыми гибкими и удобными, но за счет этого страдает безопасность.

Преимущества:  
1. Удобство для пользователя

2. гибкость

3. не требуется много усилий для поддержки и обслуживания

Недостатки:  
1. Не особо безопасны(проблема троянского коня)

2. сложно контролировать поток данных и права доступа

3. в больших системах сложно отследить кто кому что разрешил

Используется в операционках (\*nix, unix-like, a mne pohui, в винде) в гугл драйве, дропбоксе, мускуле, постгресе

Использовать для пк, в небольших компаниях и гибких средах, для норм безопасности комбинируют с мак и рбак

**Мак**

в целом очень похожа на БЛМ, используется в госструктурах и чем-то, где уровни секретности пиздец важны

существует иерархия уровней доступа, объекты с уровнем секретности (допускается повышение уровня но не понижение), субъект с уровнем доступа

как правило используется в совокупности с другими моделями безопасности тк

При проверке правомочности доступа субъекта к объекту согласно мандатной модели возможны следующие комбинации:

1. Мандатная метка субъекта равна мандатной метке объекта. В этом случае субъекту разрешено читать и изменять объект.
2. Мандатная метка субъекта выше мандатной метки объекта. Субъекту разрешено только читать объект: он его видит, но не может изменить.
3. Мандатная метка субъекта ниже мандатной метки объекта. Субъекту формально разрешено создать объект с более высокой мандатной меткой (так называемое «повышение уровня секретности объекта»). На практике у субъекта нет технической возможности для выполнения данной операции (он просто «не видит» изменяемый объект, например, файл или каталог с файлами).

Также существуют категории для горизонтального разграничения прав доступа (например, между отделами)

Ограничения и особенности:  
1. Пользователи не могут определять доступ субъектов к объектам, этим занимается только администратор

2. если пользователь хочет изменить метку объекта, и он является автором, то он должен передать метку по наследству, просто так он ее придумать не может

3. тк используется с другими моделями безопасности то могут быть коллизии и будет сложно понять в каком именно слое произошел отказ доступа

4. нужен нормальный регламент чтобы было понятно откуда и куда берутся метки и где их предоставлять

5. сложности настройки и реализации, НЕ ИСПОЛЬЗУЕТСЯ В КЛАССИЧЕСКИХ ПРИЛОЖЕНИЯХ ПОЖАЛУЙСТА ТОЛЬКО ДЛЯ СУПЕРПУПЕРГОСПРОЕКТОВ

Недостатки:  
1. Отсутствие гибкости ВООБЩЕ

2. требует очень тонкой настройки

3. проблемы с масштабированием

4. неудобство для пользователя

5. не используется в коммерции

6. очень дорого

Используется госструктурами, банками, IT гигантами с ооооочень высокими требованиями к безопасности (гугл, амазон веб сервисы), росатом и другие серьезные ребята

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Автоматически созданное описание

**белл лападула**

конкретная модель основанная на мак, со специализацией для госсекретов

*Концепт:*  
Субъектам с низкими уровнями безопасности не позволяется смотреть объекты из высокого уровня безопасности (правило нет чтения вверх). не позволяется размещать информацию в объектах более низкого уровня безопасности (второе правило - нет записи вниз). Во время обращения пользователя к информационному ресурсу сначала проверяется его уровень секретности. Следующим шагом сравниваются выданные права пользователя с уровнем секретности информационного ресурса. Если он соответствует, то субъекту будет предоставлен доступ.

недостатки: сложность реализации, случаи, когда объекты были одного уровня секретности, а стали другого - из этого рождаются правило сильного спокойствия и правило слабого спокойствия (сильного спокойствия: уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции, но теряется гибкость системы, слабого спокойствия: уровни безопасности субъектов и объектов никогда не меняются в ходе системной операции таким образом, чтобы нарушить заданную политику безопасности. Это правило может потребовать, чтобы субъекты и объекты воздерживались от действий в период времени, когда меняются их уровни безопасности)

уязвимость – утечка данных вследствие

1. Модель запрещает чтение данных с более высокого уровня но не предотвращает чтение данных с низкого уровня
2. Злоумышленник может изменить данные с более низкого уровня
3. Жесткая иерархия уровней, нельзя изменить права доступа

Основные правила: no read up; no write down

основная цель: конфиденциальность

сейчас ее используют системы минобороны США например



**биба**

была введена для улучшения модели белл лападула, но вместо уровней безопасности вводятся уровни целостности и соответственно аксиомы целостности:

***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 об*ъ*екта, характеризуемого более *ни*зким 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 к*о*торому ограничивается не*к*оторым 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. В то же время модель сохраняет все недостатки, присущие модели Белла-ЛаПадулы (потенциальная возможность организации скрытых каналов передачи информации и т.д.)

Уязвимости:

1. Утечка информации вследствие чтения данных с более высокого уровня
2. Повреждение данных при записи на более низкий уровень
3. Нет изменения прав доступа

Основная цель: обеспечивает целостность данных

Ее используют СУБД например оракл и майкрософт скл сервер, а также erp-система SAP (система по управлению бизнес процессами)

Изображение выглядит как текст, диаграмма, снимок экрана, линия

Автоматически созданное описание

**кларк вилсон**

модель существенно отличается от модели биба и от модели белл лападула. созданию способствовал анализ методов управления организациями целостностью ресурсов в неавтоматизированном офисе и была принята попытка переноса этих методов в программную среду. Получившаяся модель целостности представляет собой руководство разработчикам и проектировщикам компьютерных систем для обеспечения целостности определенных вычислительных ресурсов.

основой для модели являются транзакции, состоящие из последовательности операций, которые переводят систему из состояния в состояние. для обеспечения корректности работы системы надо поручать выполнение операций, составляющих транзакцию, нескольким субъектам, поддерживая за счет этого принцип разделения обязанностей.

Че по сути:

Есть два непересекающихся подмножества: cdi – ограниченные элементы данных(соответствуют определенным правилам и политикам) и udi – неограниченные элементы данных(не находятся под строгим контролем системы, могут быть неполными некорректными или непроверенными).

Есть девять правил, на которых стоит модель:  
1. В системе должны иметься процедуры утверждения целостности (ivp), утверждающие целостность любого cdi. (например с помощью контрольной суммы хранимой информации)

2. применение любой процедуры преобразования (ПП) к любому cdi должно сохранять целостность этого cdi

3. только ПП может вносить изменение в cdi (Это обеспечивает замкнутость в пределах набора CDI.)

4. Субъекты могут инициировать только определенные ПП над определенными CDI(правило создает квм тройки, Каждая такая тройка определяет возможность данного субъекта применить данную ПП к данному CDI. Например, если (s, t, d) является элементом отношения, то субъекту s разрешается применить ПП t к CDI d. Если же эта тройка не является элементом отношения, то такой тип применения ПП будет запрещен. Это правило гарантирует, что всегда можно определить, кто может изменить CDI и как это изменение может произойти.)

5. Тройки должны проводить соответствующую политику разделения обязанностей субъектов (не позволять субъектам изменять cdi без соответствующего вовлечения других субъектов, это предотвращает субъектов от возможности нанести ущерб целостности cdi)

6. некоторые специальные ПП могут превращать UDI в CDI

7. каждое применение ПП должно регистрироваться в специальном CDI, в который может производиться только добавление данных для создания общей картины изменений

8. система должна распознавать субъекты, пытающиеся инициировать ПП(предотвращение механизма атак, где один CDI пытается выдать за другого)

9. изменения в списках авторизации доступны только специальным субъектам (ну стандарт безопасность чтоб непонятно кто не менял)

Тк модель основана на проверенных временем бизнес методах то стоит воспринимать ее как комплекс существующих методов.

Существенным недостатком модели является то, что ее сложно реализовать в реальных компьютерных системах, потому что модель неформальная.

Уязвимости

1. Сложность реализации
2. Зависимость от правил целостности которые нужно задавать заранее
3. Неудобна в больших системах
4. Требует строгого разделения обязанностей
5. Журнал аудита может стать слабым местом и пизда

Используется оракл финансами, амазоном, ибеем, САП

**Рбак**

Управление доступом на основе ролей

Идея заключается в том, чтобы пользователь был ограничен необходимым минимумом (принцип наименьших привилегий)

У каждой роли есть набор того, что она может делать, у человека может быть несколько ролей.

Часто ставят ограничения на то, чтобы у человека не было противоречащих ролей.

Плюсом является простота администрирования

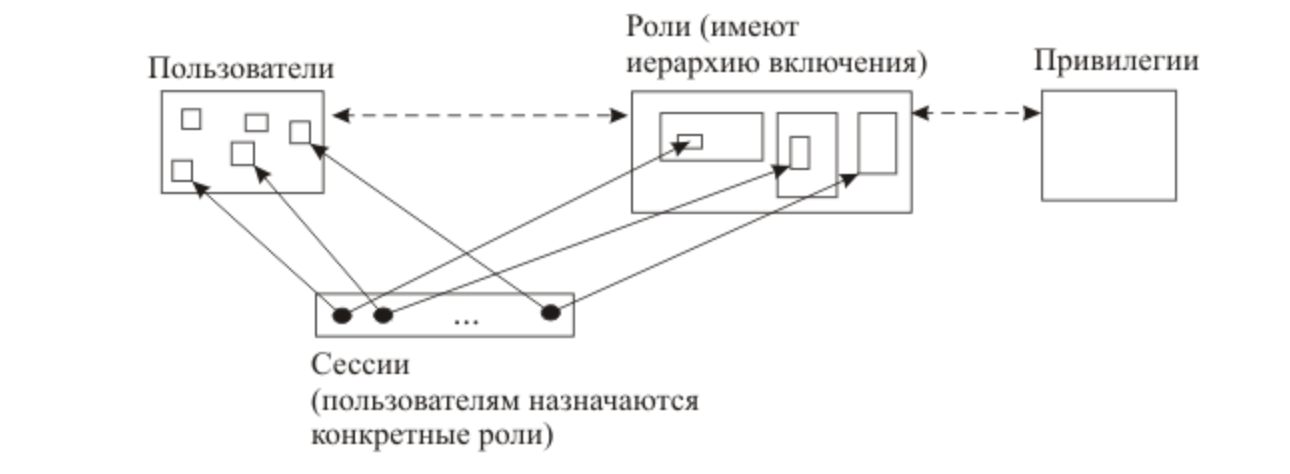
Недостатки:   
отсутствие гибкости (если нужны уникальные права то придется усложнить систему) из этого вытекает еще и проблема с большим количеством ролей в системе

Нет контекстно зависимого доступа (не учитываются внешние условия(время местоположение устройство))

Проблема унаследованный прав в пирамиде ролей

Отсутствие отслеживания действий

Используется в среднем большинством перечисленных выше компаний: Microsoft Active Directory, SELinux, FreeBSD, Solaris, СУБД Oracle, PostgreSQL 8.1, SAP R/3

****

**Сандбоксинг(песочница)**

Ну об этой модели слышали все, хотя я бы не назвала ее полноценной моделью, а только отличным дополнением к существующим моделям.

По своей сути песочница – изолированная среда для запуска непроверенного ПО без риска для основной системы.

Существует два варианта песочницы: традиционная и облачная

Суть работы песочницы:   
1. проводится сигнатурный анализ

2. источник документа проверяется по антиспам и антибот базам

3. ПО запускается в песочнице: там создается временная виртуальная машина, которая проверяет программу на подозрительную активность

4. если во время эмуляции замечена подозрительная активность, то песочница запускает ПО еще несколько раз с более расширенным функционалом проверки, чтобы в этом убедиться

5. отдельным пунктом хочется выделить ситуации, в которых вирус проявляется не сразу. Для таких моментов песочницы умеют ускорять время на ВМ.

Преимущества песочницы:  
1. Высокий уровень защиты от неизвестных угроз

2. изоляция опасных процессов

3. безопасный анализ ПО без риска заражения

4. совместимость с другими моделями безопасности

Недостатки песочницы:  
1. Современные вирусы умеют определять, что они в песочнице, так что работает не всегда(

2. ресурсоемкость, в тч и финансовая

3. ложные срабатывания

4. не защищает от всех типов атак

5. сложность настройки и управления

Используется да почти во всех технологических гигантах и просто в крупных компаниях. У каждого из вас стоит автоматическая песочница на устройстве, будь то смартфон или ноутбук, поэтому вы наверное и слышали о песочнице до того, как я вам рассказала, в отличие от других моделей безопасности

**Абак**

Разграничение доступа на основе атрибутов

У каждого объекта, субъекта, действия и среды есть свои атрибуты, на этом и строится подход: у каждой ситуации есть свои атрибуты, сравнивая которые система решает давать ли доступ к объекту.

Намного легче администрировать систему, не происходит многомерного роста условий.

Представления бизнес-правила в виде набора условий удобно использовать для фильтрации данных. Часть условий можно вычислить еще до обращения к ресурсу, а оставшиеся условия становятся фильтром для выбора данных.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Короче абак просто удобнее тк является более гибким. Сейчас он используется чаще.

Преимущества:   
1. Гибкость

2. динамическое управление доступом

3. масштабируемость

4. контекстно-зависимые правила (в отличие от рбак)

5. снижение администрирования

Недостатки:   
1. Требует доп инфраструктуры для управления атрибутами

2. медленная производительность (особенно если атрибутов много)

3. из за динамичности правил сложнее отслеживать аудиты

4. возможность создания конфликтов при переусложнении

5. требует обновления атрибутов при устаревании (постоянный контроль)

Используется платежными системами, облачными провайдерами (гугл клауд платформ, амазон веб сервисы), сап (для динамического контроля доступа к фин.данным), чутка юзается гос системами где доступ зависит от атрибута

**зиро траст**

является наиболее популярной концепцией на данный момент.

«никогда не доверяй, всегда проверяй»

В концепции фигурирует пять основных «частей»: данные, сети, нагрузка, пользователи, устройства

Приоритеты модели:

1. Защита данных
2. Необходимо сделать перемещение по сети наиболее сложным
3. Люди являются самым слабым звеном – нужно ограничивать, отслеживать и навязывать принципы получения доступа к данным
4. Под нагрузкой воспринимается весь стек используемых технологий – проверка безопасности должна быть на каждом этапе
5. Устройства также являются слабым звеном, поэтому их нужно отслеживать и сегментировать
6. Используется по большей части автоматизация, тк человек не способен уследить за таким количеством событий



Основные принципы:  
1. Требуется подтверждение для доступа к любым ресурсам «Вы должны рассматривать каждую попытку доступа к вашей сети как угрозу до тех пор, пока не подтверждено обратное, независимо от вашей модели хостинга и того, откуда происходит подключение.»

2. использовать модель наименьших привилегий и контролировать доступ: необходимо ограничивать права доступа каждого до уровня, который необходим для выполнения служебных обязанностей, для этого рекомендуют использовать рбак, потому что ее хватает и для ограничения доступа, и для того, чтобы владельцы бизнеса могли управлять разрешениями

3. отслеживать все: принципы модели нулевого доверия в том, чтобы проверить вооооооообще все перед тем, как дать доступ. Также проводится логирование и аналитика безопасности данных для отслеживания.

Доп инфа: в процессе работы пользователя его данные постоянно перепроверяются.

Для модели зиро траст даже двухфакторная аутентификация считается слабой и рекомендуется использовать многофакторную аутентификацию

Преимущества:  
1. Повышенная защита от внутренних и внешних угроз

2. гибкая безопасность

3. минимизация ущерба при взломах (спасибо микросегментации)

4. соответствие гос стандартам ес, сша, рф

5. адаптивная аутентификация

6. экономия ресурсов (траты на ликвидацию последствий атак огромны у больших компаний)

7. упрощение управления доступом (можно выдать временный доступ, отозвать права в один клик)

8. ориентирована на развитие в будущем, планируется внедрение ИИ

Недостатки:

1. Высокая сложность внедрения
2. Дороговизна (решение вопроса обеспечитвают майкрософт зиро траст и гугл бейонд корп и это требует подписки)
3. Проблемы с пользовательским опытом, тк постоянно нужно подтверждать подлинность
4. Ложные срабатывания защиты от взлома при аномальном поведении
5. Требуется постоянное подключение к интернету для проверки данных
6. Старые приложения не поддерживают современные протоколы аутентификации (например сап r/3, 1С)
7. Чем сложнее правила, тем выше шанс ошибки
8. Требования к устройствам (старые просто не работают с такими технологиями)
9. Юридические конфликты (например в россии есть закон, обязывающих хранить данные локально, но большинство зиро траст решений используют облака в сша (хотя если есть желание то варианты российских компаний тоже есть))
10. Человеческий фактор (происходят утечки данных при использовании личных почт, делятся токенами при аутентификации и тп)

Используется в гугл, майкрософт, убер, минобороны сша, нетфликс, кока кола, тесла, спотик

**будущие:**

**ии дривен контрол в тч селф хилин**

понятно, что это модель, которая управляется иишкой.

Необходимость в такой модели объясняется тем, что нападающие используют ИИ для создания вирусов и обхода защит. ИИ позволяет вирусам мутировать, адаптироваться, запускать координированные атаки, взаимодействовать с людьми как человек для повышенного успеха.

Также сейчас идут активные внедрение самовосстанавливающихся систем на основе ИИ. В целом, идея не новая, первые разработки появились еще в 00х в компании IBM, однако наиболее близко к грамотной реализации мы подобрались только сейчас и разработки в этой сфере идут крайне активно.

Работа с ИИ позволяет предупреждать атаки, когда работа с человеком основана на ликвидации последствий, что обходится намного дороже. Так еще и ликвидация с помощью ии обходится дешевле.

Как правило, ии вплетают в зиро траст модели, так проще + можно подключить модель на основе поведения в дополнение.

ИИ и безопасность постепенно становятся неразлучными. Суть стратегии при интеграции ии в модель безопасности состоит в том, чтобы

1. Ии использовался для выявления уязвимостей еще на этапе разработки
2. Ии изначально вплетен в систему, а не добавляется после всего
3. По мере развития системы, ии отслеживает и адаптирует протоколы безопасности

Преимущества такой модели:  
1. Обнаружение сложных и неизвестных угроз (0-day атаки)

2. автоматическая блокировка подозрительных действий

3. адаптивность

4. масштабируемость

5. снижение ложных срабатываний за счет контекстного анализа

6. быстрое восстановление в сравнении с человеком

Недостатки:   
1. Зависимость от качества данных

2. высокие вычислительные затраты

3. нейросети тупят и часто придумывают (ну вы заметили), поэтому не всегда могут пояснить, почему заблокировали доступ

4. что будет если ИИ взломают и дадут не те инструкции?

5. очень дорого

6. сложность интеграции в устаревшие системы

Вывод: ии не замена человеку (по крайней мере сейчас), модель нужно дорабатывать и проверять постоянно, потому что есть и плюсы, и минусы. + не стоит доверять полноценно, ИИ нуждается в постоянном обучении

Эта модель уже потихоньку внедряется в гугл, майкрософт, виза, тесла, сименс, агенство нац. Безопасности сша, сбер клауд и другие

**Изображение выглядит как диаграмма, снимок экрана, линия

Автоматически созданное описание**

**вопросики:**1. Вопрос по шестому правилу модели кларка вилсона: какие именно процедуры преобразования могут превращать UDI в СDI

ответ: например проверка и добавление данных в бд (получили данные от пользователя, преобразовали, добавили в бд – теперь данные подконтрольны системе), импорт данных из источников(пришли из внешнего файла, проверили, почистили от лишнего и добавили)

2. вопрос по модели мак: мандатная метка и мандатный уровень - чем они отличаются

Ответ: вообще это значение уровня доступа субъекта или уровень секретности объекта, отличаются тем, что мандатный уровень определяет только ранг, а мандатная метка определяет еще и полный набор атрибутов безопасности, например если бы у пекарен была МБ мак то было бы так: мандатный уровень «выпечка», мандатная метка «выпечка (вольчек 2025)»

3.чем отличается традиционная и облачная песочница

Ответ: традиционная это покупка настройка и содержание песочницы ложится на компанию, облачная это просто настройка подписки на сервисы, которые обеспечивают грамотную работу песочницы

4.как работает ускорение времени в песочнице?  
ответ: подмена ответов при вызове системного времени, основная сложность в том, чтобы подменить все значения, связанные с временем: выполнение процессов, задержки и тп, чтоб вирус не заметил, что вы ускорили выполнение.

5. что такое 0-day атаки?  
ответ: это неустраненные уязвимости или вирусы, для которых еще не разработаны защитные механизмы.

1. В чем заключается проблема троянского коня? (ну хз мб очевидный момент, но решила выделить в вопрос)  
   ответ: в том, что пользователь может выдать права кому угодно – в тч и вредоносной программе, которая маскируется под безопасную.
2. Что такое аудиты

Ответ: грубо говоря объект, в котором записываются действия в системе

1. я упоминала модель на основе поведения – что это?  
   ответ: проверка на «корректность» поведения пользователя, допустим пользователь работал обычно 5/2 с 9 до 18, а тут зашел в систему в вс в 3 часа ночи. Система обнаружит эту «подозрительную активность» и сообщит. Собственно, с таким подходом каждый из вас сталкивался)