DOI 10.37882/2223-2966.2023.6-2.40

ОЦЕНКА БЕЗОПАСНОСТИ WINE С ИСПОЛЬЗОВАНИЕМ МЕТОДОЛОГИИ STRIDE: МАТЕМАТИЧЕСКАЯ МОДЕЛЬ

WINE SAFETY ASSESSMENT USING STRIDE METHODOLOGY: A MATHEMATICAL MODEL

A. Uymin
I. Morozov

Summary: This article takes a detailed look at the security of applications running under Wine in Linux, with the main focus on discretionary and credential-based access control models. The article develops a mathematical model of both approaches and presents concrete applications of these models in the context of Wine. The article applies the STRIDE methodology to security threat analysis in Wine. The combination of a detailed security analysis and practical application of access control models makes this article useful for IT security researchers, system administrators and application developers working with Wine.

Keywords: Wine, Linux, security, discretionary access control (DAC), Mandated access control (MAC), STRIDE methodology, threat analysis, access control model, vulnerabilities.

Уймин Антон Григорьевич
старший преподаватель,
РГУ нефти и газа (НИУ) имени И. М. Губкина, г. Москва
au-mail@ya.ru
Морозов Илья Михайлович
учебный мастер
РГУ нефти и газа (НИУ) имени И. М. Губкина, г. Москва
morozowilui@gmail.com

Аннотация. Статья рассматривает систему безопасности приложений, работающих под управлением Wine в Linux, с основным фокусом на дискреционных и мандатных моделях управления доступом. В статье разрабатывается математическая модель обоих подходов и представляется конкретное применение этих моделей в контексте Wine. Статья применяет методологию STRIDE для анализа угроз безопасности в Wine. Сочетание подробного анализа системы безопасности и практического применения моделей управления доступом делает эту статью полезной для исследователей в области безопасности информационных систем, администраторов систем и разработчиков приложений, работающих с Wine.

Ключевые слова: Wine, Linux, безопасность, дискреционное управление доступом (DAC), мандатное управление доступом (MAC), методология STRIDE, анализ угроз, модель управления доступом, уязвимости.

Технологическая независимость в сфере системного программного обеспечения — это стремление страны или организации минимизировать свою зависимость от иностранных технологий и поставщиков программного обеспечения [1]. Технологическая независимость может быть достигнута путем внедрения и использования свободного и открытого программного обеспечения (FOSS) [2], в том числе операционных систем на базе Linux, и развития собственных национальных технологий.

Китай — один из основных примеров страны, уже более 10 лет движущейся в этом направлении. Один из важных шагов в этом процессе был внедрение своей собственной операционной системы на базе Linux, такой как Kylin [3]. Переход на новую ОС столкнулся с проблемой совместимости с программным обеспечением Windows, которое широко используется в китайском бизнесе и государственных структурах. Китайские специалисты приняли решение о применении Wine [4], для решения поставленной задачи. Wine был интегрирован в Kylin и другие китайские дистрибутивы Linux, чтобы обеспечить совместимость с приложениями Windows. Это позволило организациям в Китае продолжать использование привычного ПО и обеспечить более гладкий переход на новую операционную систему. Опыт Китая показывает, как можно работать в переходный период, при разработке собственной технологической платформы.

Wine — представляет собой слой совместимости, позволяющий запускать Windows-приложения на других операционных системах, таких как Linux, macOS и BSD. Имя «Wine» является рекурсивным акронимом и расшифровывается как «Wine Is Not an Emulator», что можно перевести как «Wine — это не эмулятор» [5]. Wine переводит вызовы Windows API, которые делает приложение, в POSIX-вызовы на лету, что позволяет интегрировать Windows-приложения в Unix-среду на десктопе. Тут и далее термин Unix-среда включает в себя решения на базе Linux. Wine не всегда может запустить Windows-приложения, т.к. некоторые из них могут используют очень специфические или сложные элементы Windows API, которые в настоящий момент не воссозданы в Unix-среде [6]. Основными преимуществами работы с Wine являются [7-9]:

1. **Вопрос затрат:** Wine позволяет пользователям Linux и других Unix-подобных систем запускать приложения Windows без необходимости приобретения лицензии на Windows. При этом не решается вопрос лицензирования приложения в самом Wine.
2. **Вопрос совместимости с существующим ПО:** Многие организации используют специализиро-

ванное программное обеспечение для Windows, которое может быть сложно или невозможно заменить. Wine позволяет этим организациям перейти на открытую операционную систему, сохраняя при этом возможность использования существующего ПО. Процесс конфигурирования и настройки требует больших временных и интеллектуальных затрат.

3. **Вопрос независимости от поставщика:** Использование Wine сокращает зависимость от конкретного поставщика программного обеспечения. Это не только снижает риск проблем, если поставщик вдруг изменит свои условия или прекратит поддержку, но и увеличивает свободу выбора в отношении будущих решений по программному обеспечению. Wine одинаково поддерживается в основных отечественных дистрибутивах, таких как ОС Альт, ROSA Linux, РЭД ОС, Astra Linux и др.
4. **Вопрос безопасности и прозрачности:** Как часть проекта FOSS, исходный код Wine доступен для изучения и аудита. Это означает, что можно быть уверенным в безопасности и надежности программного обеспечения, поскольку любые уязвимости или проблемы могут быть быстро обнаружены и исправлены сообществом. Необходимо отметить что данный вопрос полностью решен только в одном Российском дистрибутиве, который полностью поддерживает свободы Linux — ОС Альт. Другие дистрибутивы поддерживают логику закрытых компонентов.

Система безопасности в Wine построена на основе тех же принципов безопасности, что и в UNIX и Linux.

1. **Принцип наименьших привилегий:** Приложения, работающие через Wine, имеют те же права, что и пользователь, запустивший Wine. Если пользователь не имеет прав администратора, то и приложение, работающее через Wine, также не будет иметь этих привилегий.
2. **Ограничение области видимости:** По умолчанию, приложениям, запускаемым через Wine, доступна только домашняя директория пользователя, запустившего приложение. Это ограничивает возможный ущерб, который могут нанести злонамеренные программы.
3. **Ограничение взаимодействия с системой:** Wine является отдельным слоем, что означает, что взаимодействие с нижележащими слоями системы ограничено.

Особенностью работы с Российскими дистрибутивами является, то, что ОС Альт, ROSA Linux, РЭД ОС старят систему безопасности на базе открытых и международных систем безопасности, в том числе SELinux. Дистрибутив Astra Linux имеет собственный закрытый инструмент PARSEC.

Рассмотрим работу Wine в рамках Дискреционной модели доступа. Дискреционный контроль доступа (Discretionary Access Control, DAC) — это метод, используемый в компьютерных системах (включая Unix и Linux), для ограничения доступа к объектам на основе идентификации тех, кто пытается получить доступ. В рамках настройки системы мы можем контролировать права доступа, используя стандартные инструменты Unix для управления правами на файлы [10]. Основными инструментами являются:

1. Установка прав на файлы и папки: Использование команд chmod и chown в Unix, чтобы установить разрешения на файлы и изменить их владельца. Важно установить разрешения так, чтобы приложение, запущенное через Wine, имело доступ только к тем файлам и папкам, которые ему необходимы для работы.
2. Запуск Wine от имени определенного пользователя: Для ограничения прав доступа приложения, запущенного через Wine, необходимо запускать Wine от имени другого пользователя с более ограниченными правами доступа. Работа с командами sudo или su в Unix.
3. Использование SELinux (инструмент AppArmor в отечественных дистрибутивах не применяется): Эти инструменты позволяют настраивать дискреционные права доступа на более тонком уровне, позволяя определить, какие действия разрешены для каждого отдельного процесса в системе. Необходимо создать профиль для Wine, который ограничит его взаимодействие с системой.

В рамках рассматриваемого вопроса в модели DAC необходимо рассмотреть следующие элементы:

1. **Субъекты (S):** Это активные сущности (обычно пользователи или процессы), которые инициируют доступ к объектам. В контексте Wine, это может быть сам Wine или любое приложение, работающее в Wine.
2. **Объекты (O):** Это пассивные сущности (обычно файлы или ресурсы), к которым субъекты пытаются получить доступ.
3. **Права доступа (A):** Это действия, которые субъекты могут предпринять с объектами. Обычно это действия типа чтения (read), записи (write), и выполнения (execute).
4. **Правила (P):** Это набор правил, определяющих, какие права доступа предоставляются субъектам к объектам. Это обычно представлено в виде матрицы доступа, где строки представляют субъектов, столбцы представляют объекты, а ячейки представляют права доступа.

В контексте Wine, система контроля доступа в Unix или Linux определяет, какие действия может выполнять приложение, запущенное в Wine, и к каким ресур-

сам оно может иметь доступ. Данная модель упрощает реальную ситуацию. В реальности могут существовать и другие механизмы контроля доступа, включая мандатный контроль доступа (Mandatory Access Control, MAC) и контроль доступа на основе ролей (Role-Based Access Control, RBAC). В реальности, дискреционный контроль доступа (DAC) в контексте приложения, работающего в Wine, зависит от двух основных компонентов: политик безопасности самого Wine и политик безопасности хостовой операционной системы (в данном случае Linux):

1. **Политики безопасности Wine:** Wine представляет собой слой совместимости, который пытается эмулировать API Windows в Linux. Поэтому приложения, запущенные через Wine, могут ожидать, что они работают в среде Windows, и полагаться на политики DAC, которые бы применялись, если они действительно работали в Windows. Однако, эмуляция этих политик в Wine может быть неидеальной, и в некоторых случаях Wine может накладывать свои собственные ограничения на доступ приложения к ресурсам. Это может включать в себя ограничения на доступ к определенным директориям, ограничения на взаимодействие с нативными Linux-процессами и т. д.
2. **Политики безопасности хостовой операционной системы:** В конечном итоге, приложение, работающее в Wine, выполняется в контексте хостовой операционной системы (Linux) и поэтому должно следовать ее политикам DAC. Это означает, что приложение может иметь доступ только к тем файлам и ресурсам, которые разрешены пользователю или процессу, который запустил Wine.

Если мы хотим представить это математически, мы могли бы ввести дополнительное разделение в нашу функцию доступа. Давайте определим следующие переменные:

— **S** — это множество всех субъектов (пользователей, процессов), которые могут запрашивать доступ к объектам. В контексте Wine, это может быть процесс Wine или приложение, работающее через Wine.
— **O** — это множество всех объектов (файлов, ресурсов), к которым могут быть запросы на доступ.
— **A** — это множество всех возможных действий (read, write, execute и т. д.), которые субъект может запросить выполнить на объект.

Теперь, мы определим две функции доступа, которые будут определять, какие действия субъект может выполнить с объектом:

1. **F_Wine: S x O -> P(A):** Эта функция определяет правила доступа в соответствии с политиками безопасности Wine. Для каждой пары субъект-объект (s, o), F_Wine(s, o) будет возвращать подмножество A, представляющее все действия, которые s может выполнить на o в соответствии с политиками безопасности Wine.
2. **F_OS: S x O -> P(A):** Эта функция определяет правила доступа в соответствии с политиками безопасности хостовой операционной системы (Linux). Для каждой пары субъект-объект (s, o), F_OS(s, o) будет возвращать подмножество A, представляющее все действия, которые s может выполнить на o в соответствии с политиками безопасности Linux.

Для определения итоговых прав доступа для приложения, работающего в Wine, мы можем использовать операцию пересечения ($\cap$) для этих двух функций:

— F: S x O -> P(A), где для каждой пары субъект-объект (s, o), F(s, o) = F_Wine(s, o) $\cap$ F_OS(s, o).

Это означает, что для каждой пары субъект-объект, итоговые права доступа будут представлять собой пересечение прав, предоставляемых политиками безопасности Wine и политиками безопасности хостовой операционной системы.

Для учета влияния одного приложения на другое в контексте Wine, можно представить каждое приложение как отдельный субъект в нашей модели контроля доступа. Это означает, что мы будем рассматривать не только взаимодействие приложений с объектами (например, файлами), но и их взаимодействие друг с другом. В этом контексте, мы можем расширить наше определение множества объектов, чтобы включить в него другие приложения. То есть, O будет включать в себя не только файлы и ресурсы, но и другие приложения, работающие через Wine. Также нам потребуется расширить наше множество действий A, чтобы включить в него возможные действия, которые приложения могут выполнять в отношении друг друга (например, «отправка сообщения», «чтение памяти» и т.д.). Функции доступа будут определяться следующим образом:

1. **F_Wine: S x O -> P(A):** Для каждой пары «приложение-объект» (s, o), F_Wine(s, o) будет возвращать подмножество A, представляющее все действия, которые приложение s может выполнить в отношении объекта o в соответствии с политиками безопасности Wine. Если o является другим приложением, то это будет включать в себя все действия, которые s может выполнить в отношении o.
2. **F_OS: S x O -> P(A):** Аналогично, для каждой пары «приложение-объект» (s, o), F_OS(s, o) будет возвращать подмножество A, представляющее все действия, которые приложение s может выполнить в отношении объекта o в соответствии с политиками безопасности хостовой операционной системы.

Окончательная функция доступа по-прежнему будет определяться как F(s, o) = F_Wine(s, o) ∩ F_OS(s, o).

Эта модель позволяет учитывать влияние одного приложения на другое в контексте Wine.

Рассмотрим работу Wine в рамках Мандатной модели контроля доступа (MAC) обычно базируется на принципе минимальных привилегий и применяется в системах с высоким уровнем безопасности. Решения ОС Альт, ROSA Linux, РЭД ОС базируются на технологическом стеке SELinux. Решение Astra Linux на отечественной закрытой платформе PARSEC [11]. Мандатная модель контроля доступа отличается от дискреционной модели контроля доступа, в которой права доступа определяются владельцем объекта. В MAC права доступа определяются центральной политикой безопасности. Для описания модели мандатного контроля доступа в контексте Wine, мы можем использовать аналогичный подход, как и для DAC:

- **S** — это множество всех субъектов (приложений, запущенных через Wine).
- **O** — это множество всех объектов (файлов, ресурсов, возможно, других приложений).
- **A** — это множество всех возможных действий (read, write, execute и т. д.), которые субъект может запросить выполнить на объект.
- **L** — это множество уровней безопасности, которые могут быть присвоены субъектам и объектам. В контексте MAC, уровни безопасности обычно упорядочены.

Определим функции:

1. **F_Wine: S x O -> P(A):** Эта функция определяет правила доступа в соответствии с политиками безопасности Wine. Для каждой пары субъект-объект (s, o), F_Wine(s, o) будет возвращать подмножество A, представляющее все действия, которые s может выполнить на o в соответствии с политиками безопасности Wine.
2. **F_OS: S x O -> P(A):** Эта функция определяет правила доступа в соответствии с политиками безопасности хостовой операционной системы (Linux). Для каждой пары субъект-объект (s, o), F_OS(s, o) будет возвращать подмножество A, представляющее все действия, которые s может выполнить на o в соответствии с политиками безопасности Linux.

В модели MAC, мы также учитываем уровни безопасности субъектов и объектов. Для этого мы определяем две функции:

- **l_S: S -> L:** Эта функция присваивает каждому субъекту уровень безопасности из множества L.
- **l_O: O -> L:** Эта функция присваивает каждому объекту уровень безопасности из множества L.

В отличие от модели DAC, в которой итоговые права доступа определяются как пересечение прав, предоставляемых политиками безопасности Wine и политиками безопасности хостовой операционной системы, в модели MAC итоговые права доступа определяются на основе уровней безопасности субъекта и объекта. В общем случае, субъект s может выполнять действие a на объекте o, только если l_S(s) ≥ l_O(o) и если это действие разрешено функциями F_Wine и F_OS. То есть:

- **F: S x O -> P(A)**, где для каждой пары субъект-объект (s, o), F(s, o) = (F_Wine(s, o) ∩ F_OS(s, o)) если l_S(s) ≥ l_O(o), и пустое множество в противном случае.

Это упрощенное описание модели MAC в контексте Wine. В реальной системе могут быть дополнительные факторы, которые могут влиять на политику безопасности, включая различные классификации информации, обязательные атрибуты безопасности, ролевые модели контроля доступа и т.д.

В настоящий момент, Российские предприятия находятся в переходном периоде, коде необходимо не прерывая бизнес-процессы, обеспечить переход на технологически независимое программное обеспечение [12]. В этом контексте есть понимание о высокой готовности системного программного обеспечения, при недостатке прикладного программного обеспечения [13]. В качестве примера можно привести решения Компас 3D. Компания Аскон представила Роадмап [14] в кортом прописала порядок перехода на отечественное ПО, представлена на рисунке 1.

Необходимо отметить, что в большинстве случаев на предприятиях основной упор делается на процесс перехода на отечественное ПО. При этом не всегда уделяется необходимое внимание решениям безопасности в переходной период. В рамках сотрудничества с одним из отечественных предприятий было принято решение рассмотреть аспекты переходного периода в разрезе вопросы информационной безопасности. Заказчик выбрал в качества методологии методологию STRIDE.

Методология STRIDE, разработанная в Microsoft, является одной из популярных методологий для идентификации и классификации угроз безопасности [15]. STRIDE является аббревиатурой, каждая буква которой обозначает определенный тип угрозы: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, и Elevation of privilege указаны в таблице 1.

Методология STRIDE обеспечивает хороший фреймворк для анализа и оценки угроз безопасности в Wine. Она позволяет систематически идентифицировать и классифицировать угрозы, а также привязывать их к конкретным свойствам системы (конфиденциальность,

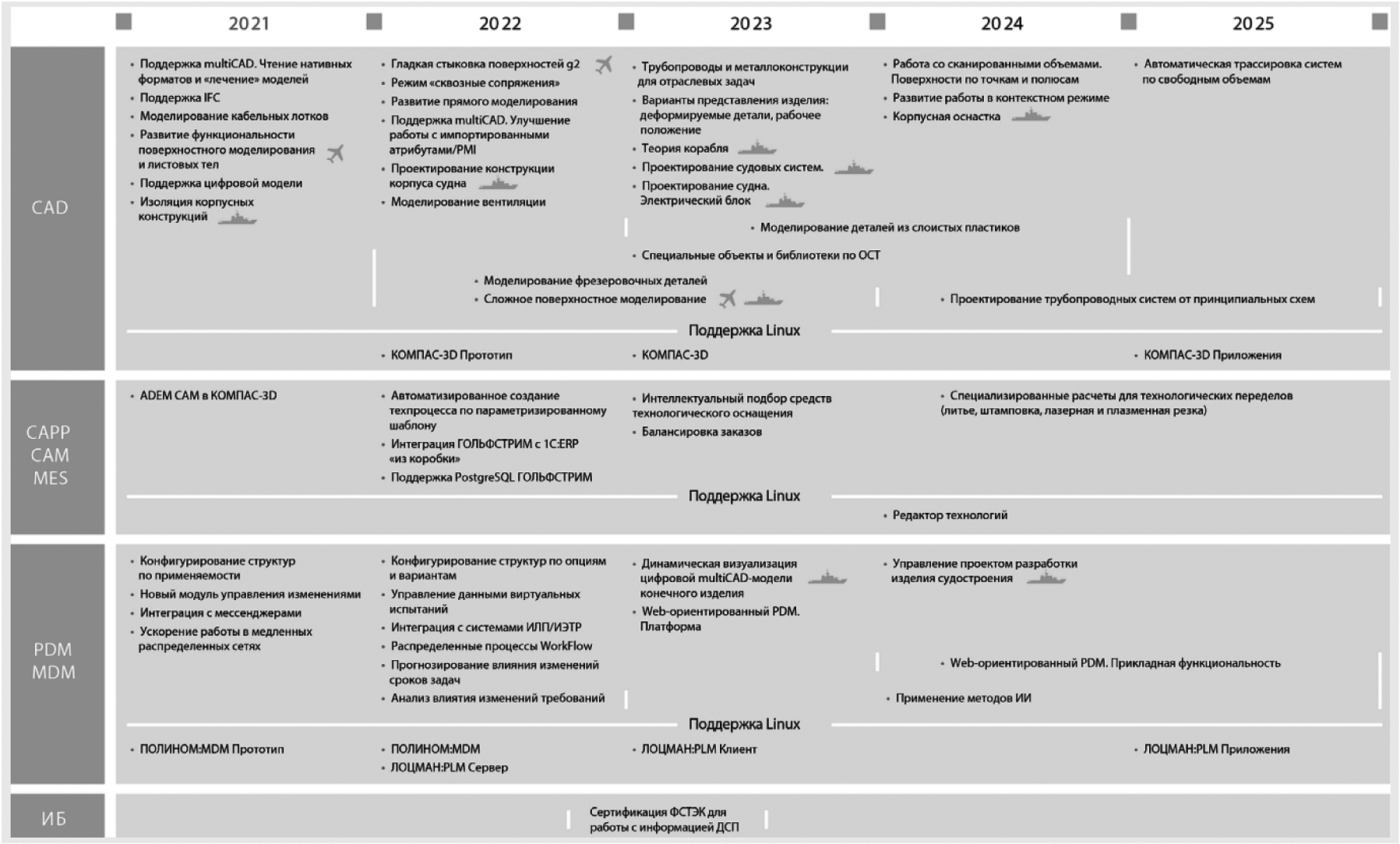

Рис. 1. Дорожная карта машиностроительной стратегии АСКОН

Таблица 1.

Методология STRIDE

Угрозы (Threats)	Описание	Свойство	Пример
Spoofing (Подделка)	Подделка идентификационных данных может произойти, когда вредоносное приложение имитирует легитимное.	Целостность	Вредоносное приложение может подделывать свой процесс или имя файла, чтобы выглядеть как легитимное приложение.
Tampering (Несанкционированное изменение)	Несанкционированное изменение может произойти, если приложение, работающее в Wine, изменяет данные, к которым у него не должно быть доступа.	Целостность	Приложение может пытаться изменить или удалить файлы, к которым у него не должно быть доступа.
Repudiation (Отказ от ответственности)	Отказ от ответственности возникает, когда действия, выполняемые приложением, не могут быть отслежены или аудитированы.	Целостность	Вредоносное приложение может удалять журналы или другие следы своих действий, чтобы избежать обнаружения.
Information disclosure (Раскрытие информации)	Раскрытие информации может произойти, если приложение имеет доступ к информации, к которой у него не должно быть доступа.	Конфиденциальность	Приложение может пытаться считать файлы или данные, которые ему не должны быть доступны.
Denial of service (Отказ в обслуживании)	Отказ в обслуживании происходит, когда приложение может вывести из строя систему или другие приложения.	Доступность	Приложение может потреблять все ресурсы системы, что приводит к замедлению или остановке работы системы.
Elevation of privilege (Повышение привилегий)	Повышение привилегий происходит, когда приложение может получить больше прав, чем ему было изначально предоставлено.	Целостность, Доступность	Приложение может эксплуатировать уязвимость в Wine или в хостовой операционной системе для получения повышенных привилегий.

целостность, доступность), что помогает более точно оценить влияние потенциальных угроз. Результаты анализа подтверждают, что Wine, как и любая другая система эмуляции или трансляции, подвержен ряду угроз безопасности. Это включает в себя не только угрозы, связанные с совместимостью и взаимодействием с хост-ОС, но и угрозы, связанные с самими запускаемыми приложениями Windows.

Однако анализ также подчеркивает важность правильной настройки и управления безопасностью Wine. Использование дискреционного и мандатного контроля доступа, а также соблюдение принципа наименьших привилегий, может помочь снизить риск многих угроз. Методология STRIDE представляет собой полезный инструмент для анализа безопасности Wine и может быть использована для улучшения стратегий обеспечения безопасности и разработки практик управления рисками. Решение апробировано на предприятии. Оценки эксплуатации будут получены через контрольный период.

ЛИТЕРАТУРА

1. Кротова М.В. Возможности методологии системного анализа применительно к разработке стратегии обеспечения технологического суверенитета России // Россия: тенденции и перспективы развития. 2022. №17–2.
2. Дорошенко Марина Евгеньевна, Скрипкин Кирилл Георгиевич Развитие национального рынка программного обеспечения: альтернативы государственной политики // Форсайт. 2013. №1.
3. Дегтерев Денис Андреевич, Рамич Мирзет Сафетович, Цвык Анатолий Владимирович США — КНР: «властный транзит» и контуры «конфликтной биполярности» // Вестник РУДН. Серия: Международные отношения. 2021. №2.
4. Киргинцев Михаил Викторович, Нечаев Сергей Александрович, Киргинцева Наталья Сергеевна Проблемы и перспективы использования отечественного программного обеспечения в военных образовательных организациях // Военная мысль. 2022. №10.
5. WineHQ. [Электронный ресурс]. — Режим доступа: https://www.winehq.org/ (Дата обращения: 24 мая 2023).
6. Вахранев А.Б. Реалии импортозамещения программного обеспечения в образовании // Современные социальные и экономические процессы: проблемы, тенденции, перспективы регионального развития. 2023. №1. URL: https://cyberleninka.ru/article/n/realii-importozamescheniya-programmnogo-obespecheniya-v-obrazovanii (дата обращения: 23.05.2023).
7. Говорухин Юрий Михайлович, Домрачев Алексей Николаевич, Криволапов Виктор Григорьевич, Палеев Дмитрий Юрьевич О фактических аэродинамических сопротивлениях горных выработок, полученных в ходе воздушно-депрессионных съёмок на угольных шахтах // Известия ТулГУ. Науки о Земле. 2022. №3.
8. Хахаев Иван Анатольевич, Кузьмин Владимир Александрович, Фогель Леонид Сергеевич, Орехов Дмитрий Андреевич, Цыганов Андрей Викторович Защита первичных ветеринарных данных при создании ГИС мониторинга эпизоотической ситуации по АЧС // Природные ресурсы, среда и общество. 2021. №3.

9. Телков Александр Юрьевич, Данилова Ольга Юрьевна, Телкова Светлана Анатольевна Методологический подход к вопросу интеллектуального переноса информации между базами данных интегрированных систем безопасности // Вестник ВИ МВД России. 2019. №4.
10. Кругликов Сергей Владимирович, Дмитриев Владимир Александрович, Степанян Арарат Баркевович, Максимович Елена Павловна Политика управления доступом в системе защиты информации высокопроизводительной системы обработки геолого-геофизических данных // Вопросы кибербезопасности. 2018. №3 (27).
11. Девянин Петр Николаевич, Леонова Мария Александровна Приёмы описания модели управления доступом Осси Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения её верификации инструментами Rodin и Prob // ПДМ. 2021. №52.
12. Уймин, А.Г. Применение отечественного программного обеспечения для перестройки образовательного процесса вуза в рамках подготовки кадров цифровизации производства / А.Г. Уймин // Цифровая трансформация промышленности: новые горизонты: Сборник научных трудов по материалам 3-й Всероссийской научно-практической конференции, Москва, 10 ноября 2022 года. Том 1. — Москва: Общество с ограниченной ответственностью «Русайнс», 2022. — С. 398–405. — EDN QECBCF.
13. Греков, В.С. Перспективы кибербезопасности в нефтегазовой отрасли / В.С. Греков, А.Г. Уймин // Губкинский университет в решении вопросов нефтегазовой отрасли России : Тезисы докладов VI Региональной научно-технической конференции, посвященной 100-летию М.М. Ивановой, Москва, 19–21 сентября 2022 года. — Москва: Российский государственный университет нефти и газа (национальный исследовательский университет) имени И.М. Губкина, 2022. — С. 1108–1109.
14. КОМПАС-3D x64 и Linux на базовом Wine 8.0 [Электронный ресурс] // Habr.com. — Режим доступа: https://habr.com/ru/companies/ascon/articles/715710/ (Дата обращения: 24 мая 2023)
15. Криштаносов Виталий Брониславович Методология оценки и управления цифровыми рисками // Труды БГТУ. Серия 5: Экономика и управление. 2021. №2 (250).