Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Информационная безопасность

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Исследование возможностей подсистемы аудита Linux (auditd) для расследования инцидентов безопасности

Студент	ИУ8-95		Б. Е. Шекунов
	(группа)	(подпись, дата)	(И.О. Фамилия)
Руководитель курсового проекта			Д. В. Ефанов
		(подпись, дата)	(И.О. Фамилия)
Консультант			
		(подпись, дата)	(И.О. Фамилия)

2025 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ8
(индекс)
М.А. Басараб
(И.О. Фамилия)
(подпись)
(дата)

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Защита в операционных системах
Студент группы ИУ8-95 Шекунов Борис Евгеньевич
(Фамилия, имя, отчество)
Тема курсового проекта Исследование возможностей подсистемы аудита Linux (auditd) для расследования инцидентов безопасности

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
практическая

Источник тематики (кафедра, предприятие, НИР) кафедра ИУ8

Задание Провести анализ возможностей подсистемы аудита Linux (auditd) для расследования инцидентов информационной безопасности. Разработать методику использования auditd для выявления различных инцидентов безопасности. Разработать программу с графическим интерфейсом для визуализации и анализа журналов аудита. Провести отладку и тестирование программы. Загрузить программу в репозиторий github.com. Оформить отчёт.

Оформление курсового проекта:
Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

Дата выдачи задания « ___ » _____ 2025 г.

Руководитель курсового проекта _____ Д.В. Ефанов
(подпись, дата) (И.О. Фамилия)

Студент _____ Б.Е. Шекунов
(подпись, дата) (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

КАЛЕНДАРНЫЙ ПЛАН
на выполнение курсового проекта

по дисциплине Защита в операционных системах
Студент группы ИУ8-95 Шекунов Борис Евгеньевич
(Фамилия, имя, отчество)
Тема курсового проекта Исследование возможностей подсистемы аудита Linux (auditd) для расследования инцидентов безопасности

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Руководитель КР	Куратор
1.	Задание на выполнение курсового проекта	_____ *Планируемая дата*			
2.	1 модуль	_____ *Планируемая дата*			
3.	2 модуль	_____ *Планируемая дата*			
4.	Оформление РПЗ	_____ *Планируемая дата*			
5.	Подготовка доклада и презентации (при необходимости)	_____ *Планируемая дата*			
6.	Защита курсового проекта	_____ *Планируемая дата*			

Студент _____ Руководитель проекта _____
(подпись, дата) (подпись, дата)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современных Linux-системах подсистема аудита auditd используется как стандартное средство регистрации событий безопасности: попыток аутентификации, операций с файлами, изменений конфигурации и действий привилегированных пользователей. Эти данные фиксируются в журнале /var/log/audit/audit.log и могут служить основным источником информации при разборе инцидентов информационной безопасности.

При этом формат журнала auditd ориентирован в первую очередь на машинную обработку: события представляются в виде набора пар «ключ=значение», одно логическое действие нередко описывается несколькими строками, а общий объём данных может быть значительным. В результате прямой анализ журнала в текстовом виде требует существенных временных затрат и усложняет практическое использование auditd в расследованиях.

В рамках данной курсовой работы рассматривается подход, позволяющий упростить использование подсистемы аудита Linux для анализа инцидентов. Цель работы заключается в исследовании функциональных возможностей auditd с точки зрения расследования событий информационной безопасности и в демонстрации практической методики работы с журналами аудита.

Для поддержки методики разработано прикладное программное средство с графическим интерфейсом – Linux Audit Viewer. Программа предназначена для визуализации и фильтрации стандартных журналов auditd, а также для полуавтоматического поиска типовых инцидентов (подбор пароля по SSH, изменения критичных файлов, запуск оболочки от сервисных пользователей) и формирования обзорной статистики. В отчёте последовательно рассматриваются основы подсистемы аудита, описывается методика расследования инцидентов, архитектура и реализация разработанного приложения, результаты тестирования и возможные направления дальнейшего развития решения.

ОСНОВНАЯ ЧАСТЬ

1 Подсистема аудита Linux как источник данных для расследования

1.1 Системный аудит и его место в расследовании инцидентов

Современные операционные системы генерируют большое количество журналов: системные логи, логи приложений, сетевые логи и т.п. Однако далеко не каждый журнал одинаково полезен именно для расследования инцидентов информационной безопасности. Для восстановления полной картины действий нарушителя требуется максимально детализированный, непротиворечивый и достоверный источник данных о том, что именно происходило на уровне системы, а не только на уровне отдельных сервисов.

Подсистема системного аудита решает эту задачу за счёт централизованного сбора событий, связанных с действиями пользователей и процессов: запуск программ, доступ к файлам и каталогам, операции с привилегиями, аутентификация, изменения конфигурации и т.п. Аудит не ограничивается сообщениями о «фатальных» ошибках или предупреждениях, а фиксирует именно факт обращения к чувствительным ресурсам и выполнения критичных действий. Это делает его ключевым источником для последующей реконструкции хронологии и моделирования сценария атаки.

С точки зрения расследования инцидентов системный аудит выполняет несколько функций:

- обеспечивает доказательную базу: по журналам можно показать, какой пользователь или процесс выполнил конкретное действие, в какое время и с какими параметрами;
- позволяет восстанавливать цепочку событий: от первичного входа на систему до изменения прав доступа, модификации конфигурации и установки вредоносного ПО;
- поддерживает контроль целостности и подотчетности: любые отклонения от нормального поведения пользователей и служб становятся наблюдаемыми и поддающимися анализу;

- служит основой для инцидент-реакции и постфактум-аналитики: без корректных журналов невозможно ни быстро локализовать угрозу, ни убедительно доказать факт нарушения.

Отдельно стоит подчеркнуть важность целостности журналов. Если логи могут быть свободно изменены или удалены злоумышленником, они перестают быть надёжным доказательством. Именно поэтому к подсистеме аудита и к месту хранения её журналов предъявляются повышенные требования по защите и контролю доступа. Нарушение целостности журналов рассматривается в современных методологиях как отдельный класс атак и само по себе может являться признаком инцидента.

Таким образом, системный аудит в целом, и подсистема аудита Linux в частности, выступают не просто вспомогательным механизмом логирования, а базовой инфраструктурой для построения методик расследования инцидентов, поддержания соответствия требованиям регуляторов и обеспечения информационной безопасности.

1.2 Архитектура и принцип работы подсистемы auditd

В Linux аудит реализован в виде отдельной подсистемы, состоящей из компонент ядра и пользовательского пространства. В ядро включён Audit Subsystem, который перехватывает определённые системные вызовы и события и формирует на их основе сообщения аудита. Эти сообщения передаются в пользовательское пространство через специальный интерфейс ядро-пользователь.

На стороне пользовательского пространства ключевую роль играет демон auditd – Audit Daemon. Он отвечает за приём сообщений от ядра, их буферизацию, окончательную обработку и запись в лог-файлы, как правило в каталог /var/log/audit/ (основной файл – /var/log/audit/audit.log). В зависимости от конфигурации auditd может вести ротацию логов, контролировать заполненность файловой системы и предпринимать действия при нехватке места (например, переход в режим только для чтения).

Архитектурно подсистема аудита включает следующие элементы:

- ядро (audit subsystem) – генерирует события, основываясь на заданных правилах аудита (audit rules);
- демон auditd – принимает события и записывает их в журналы;
- подсистема диспетчеризации audispd / плагины – передаёт события в реальном времени внешним аналитическим системам (SIEM, IDS, средства корреляции и т.п.);
- утилиты администрирования – auditctl/augenrules для управления правилами, ausearch/aureport для поиска и формирования отчётов по журналам.

Правила аудита определяют, что именно нужно логировать: доступ к конкретным файлам и каталогам, вызовы определённых системных функций, действия конкретного пользователя, попытки аутентификации и т.д. Эти правила задаются в конфигурационных файлах (например, rules.d, audit.rules) и загружаются в ядро при старте системы или демона.

Важно, что часть логики подсистемы работает на уровне ядра: это позволяет фиксировать события с достаточно высокой степенью детализации и независимости от приложений. Если действие инициирует даже привилегированный процесс, оно всё равно будет проходить через системные вызовы, перехватываемые подсистемой аудита. Это делает аудит ценным инструментом для контроля за действиями как обычных пользователей, так и администраторов или сервисных процессов.

Таким образом, архитектура auditd обеспечивает тонко настраиваемый, низкоуровневый и централизованный сбор событий, который далее может использоваться как стандартными консольными утилитами (ausearch, aureport), так и внешними аналитическими инструментами, включая разработанное в данной работе приложение Linux Audit Viewer.

1.3 Формат журналов auditd и ключевые типы событий

Журналы подсистемы аудита по умолчанию сохраняются в файл /var/log/audit/audit.log (и его ротации) и представляют собой текстовые записи, каждая из которых состоит из набора пар ключ=значение, дополненных служебной информацией.

Типичная строка имеет вид:

type=SYSCALL msg=audit(1713809123.123:456): arch=c000003e syscall=59 success=yes ...

В начале строки указывается поле type=..., определяющее тип записи (например, SYSCALL, PATH, USER_AUTH, USER_LOGIN и др.). Далее следует блок msg=audit(…:ID), содержащий временную метку события (в секундах и долях секунды от эпохи Unix) и идентификатор события (event ID). Этот идентификатор связывает несколько физических строк, относящихся к одному логическому событию.

Особенность журналов auditd заключается в том, что одно событие часто представлено несколькими записями разных типов, объединённых общим идентификатором. Например, при выполнении системного вызова execve демон может сгенерировать:

- запись SYSCALL – общая информация о системном вызове (PID, UID, AUID, имя команды, путь к исполняемому файлу, флаг успешности);
- одну или несколько записей PATH – пути к файлам, задействованным в операции (исполняемый файл, конфиги, библиотеки и т.д.);
- запись CWD – текущий рабочий каталог процесса;
- дополнительные записи других типов, если это предусмотрено правилами.

Для расследования инцидентов наибольший интерес представляют следующие классы событий:

- события аутентификации и управления учетными записями – USER_AUTH, USER_LOGIN, LOGIN, USER_ACCT и др.;

- системные вызовы и операции с файлами – SYSCALL, PATH, CWD, EXECVE и т.п.;
- события изменения конфигурации и политик – записи, связанные с изменением правил аудита, параметров безопасности и т.д.;
- события, связанные с сетью – записи, содержащие поля с адресами (addr, addr4, addr6), портами, протоколами.

Каждая запись содержит детализированный набор полей: идентификаторы процессов (PID, PPID), идентификаторы пользователей (UID, EUID, AUID), имя команды (comm), путь к исполняемому файлу (exe), коды возвращаемых значений, флаг success=... и др. Наличие полей вроде key=..., задаваемых в правилах аудита, позволяет помечать события как относящиеся к определённой политике (например, -k sshd_config при мониторинге конфигурации SSH).

С практической точки зрения формат журналов auditd обладает одновременно сильной стороной и серьёзным недостатком:

- с одной стороны, записи очень структурированы и подробны, что значительно облегчает машинный анализ, корреляцию событий и построение отчётов;
- с другой стороны, для человека «сырые» логи трудны для прямого чтения, особенно когда одно действие разбито на несколько строк. Без специализированных инструментов (таких как ausearch, aureport или внешние визуализаторы) расследование по текстовому файлу audit.log становится трудозатратной задачей.

Именно это обстоятельство оправдывает разработку прикладных средств, подобных Linux Audit Viewer: они опираются на стандартный текстовый формат журналов auditd, но предоставляют поверх него удобный уровень представления данных – с агрегацией строк, фильтрацией по ключевым полям и визуализацией типовых сценариев инцидентов. В дальнейшем в отчёте будет показано, как такая надстройка позволяет существенно упростить реализацию методики расследования инцидентов, описанной в последующих главах.

2 Методика расследования инцидентов с использованием auditd

2.1 Общая схема и этапы расследования

Методика расследования инцидентов на базе подсистемы аудита Linux (auditd) опирается на стандартный жизненный цикл инцидента: подготовка, обнаружение, анализ, выводы и меры по предотвращению повторения. auditd в этой схеме выступает основным источником событий низкого уровня – системных вызовов, операций с файлами, попыток аутентификации и т.п., пригодных для последующего постфактум-анализа.

Практически процесс можно разбить на несколько этапов:

1. Подготовка (preparation).

На этом этапе включается и настраивается подсистема аудита: определяются правила, какие действия и объекты нужно логировать, конфигурируется место хранения журналов, ротация и контроль заполнения диска. Правильная настройка на этом шаге критична: если нужные события не попали в журнал – восстановить картину инцидента будет невозможно.

2. Фиксация инцидента / подозрительной активности.

Сама по себе подсистема аудита не «знает», где инцидент, а где нормальное поведение: она просто пишет события. Сигналом к расследованию становится внешнее событие – срабатывание другого средства защиты, жалоба пользователя, обнаружение аномалий в сервисе, подозрительная запись в auth.log и т.п. На этом шаге формулируется первичный вопрос: *что именно мы хотим проверить в логах auditd* (подбор пароля, изменение конфигурации, запуск оболочки и т.д.).

3. Сбор и предварительная обработка журналов.

Для анализа отбирается журнал /var/log/audit/audit.log (и, при необходимости, файлы ротации) за интересующий период. На практике на этом этапе часто используют стандартные утилиты ausearch и aureport для первичного поиска по ключам, видам событий, пользователям и временным интервалам.

В рамках данного проекта эта же задача решается через программу с GUI: журнал загружается в Linux Audit Viewer, где «сырые» строки конвертируются в структурированные события с единым форматом полей.

4. Детальный анализ и реконструкция хронологии.

Основной этап методики. Аналитик:

- выделяет интересующий временной интервал;
- отбирает события по типу (например, только аутентификация или только системные вызовы с изменением файлов);
- группирует события по пользователям, IP-адресам, процессам;
- связывает строки с одинаковым идентификатором события (event ID) в единое логическое действие;
- по цепочке действий восстанавливает ход атаки или несанкционированных действий.

5. Формирование выводов и рекомендаций

На основе полученной хронологии формулируются ответы на практические вопросы:

- каким способом был реализован доступ;
- какие учетные записи и сервисы были задействованы;
- какие файлы и конфигурации изменены;
- есть ли признаки попыток скрыть следы (отключение auditd, чистка логов).

Далее вырабатываются меры реагирования и предотвращения повторения: изменение настроек аудита, усиление контроля доступа, донастройка мониторинга и т.п.

Разработанное приложение Linux Audit Viewer вписывается в эту схему как инструмент для этапов 3–4: оно не заменяет саму методику расследования и не подменяет auditd, а упрощает человеку работу с журналами – преобразует их в удобный вид, помогает отбирать события и автоматически реализует часть типовых сценариев поиска.

2.2 Настройка auditd под задачи расследования

Чтобы методика работала на практике, необходимо заранее продумать, какие именно события должны фиксироваться в журналах. auditd позволяет достаточно гибко задавать правила: мониторить отдельные файлы, каталоги, системные вызовы, действия конкретных пользователей или групп.

В базовом варианте для целей расследования инцидентов целесообразно обеспечить как минимум:

- Логирование аутентификации и входов в систему

Отслеживаются события входа по SSH и локальной аутентификации, изменения учётных записей, применение sudo. Это могут быть как записи auditd (USER_AUTH, USER_LOGIN и др.), так и параллельный анализ auth.log. Такие данные нужны для расследования подборов пароля, угонов учётных записей и несанкционированных входов.

- Мониторинг критичных файлов и конфигурации

Настраиваются правила на файловые объекты, изменение которых потенциально критично: файлы паролей и теневых паролей (/etc/passwd, /etc/shadow), конфигурация SSH (/etc/ssh/sshd_config), конфиги средств безопасности, конфигурация самого auditd и его правил. Рекомендуется снабжать такие правила ключами (-k critical_conf, -k auth_files и т.п.), чтобы затем можно было быстро отфильтровать соответствующие события.

- Отслеживание запусков программ и оболочек

Полезно логировать системные вызовы execve для ключевых бинарников: системных оболочек (/bin/bash, /bin/sh), интерпретаторов (python, perl, php) и при необходимости – утилит администрирования. В сочетании с информацией о пользователе, PID и текущем каталоге это позволяет выявлять подозрительные запуски оболочки от сервисных аккаунтов (например, от имени веб-сервера).

- Контроль за самим механизмом аудита

Важно отслеживать изменения конфигурации auditd, его правил, попытки остановить демон и т.п. Отдельные методики прямо рекомендуют правила для мониторинга файлов /etc/audit/auditd.conf, /etc/audit/rules.d/*.rules и событий типа

CONFIG_CHANGE, поскольку их изменение может означать попытку «ослепить» систему и скрыть следы.

Помимо содержания правил, методика учитывает и организационные моменты:

- Журналы auditd должны храниться достаточное время для типичных сценариев расследования и иметь корректно настроенную ротацию.
- Желательно предусмотреть защиту логов от несанкционированного удаления и изменения (например, перенос в удалённое хранилище, использование неизменяемых файловых систем или центральных SIEM/лог-платформ).

В рамках разработки Linux Audit Viewer предполагается, что указанные правила уже настроены, а на вход в программу подаётся обычный текстовый журнал audit.log, сформированный в соответствии с этими правилами. Приложение не занимается генерацией правил и не заменяет этап подготовки; оно работает с тем, что auditd уже записал.

2.3 Анализ журналов и типовые сценарии инцидентов

Ключевая часть методики – описание того, как по журналам auditd выявлять конкретные типы инцидентов. В рамках курсовой работы детально рассматриваются три базовых сценария, которые одновременно:

- часто встречаются в практических кейсах;
- хорошо ложатся на модель событий auditd;
- реализованы в виде сценариев поиска в разработанном приложении.

1. Подбор пароля по SSH

Цель анализа: выявить многократные неуспешные попытки аутентификации по SSH за короткий промежуток времени, исходящие от одного IP-адреса или направленные на одного пользователя. Такие цепочки событий являются типичным признаком подбора пароля.

На что смотреть в auditd:

- события аутентификации и логина (USER_AUTH, USER_LOGIN и др.);

- поле успешности операции (success=no);
- указание процесса /usr/sbin/sshd или соответствующего сервиса в полях exe/comm;
- IP-адрес источника в полях addr, addr4, addr6 или связанных записях.

Шаги анализа:

1. Ограничить временной интервал вокруг подозрительного периода (по сигналу SIEM, по времени жалобы и т.п.).
2. Отобрать неуспешные попытки входа по SSH.
3. Сгруппировать события по паре *(пользователь, IP-адрес)*.
4. Для каждой группы оценить количество неуспешных попыток в скользящем временном окне (например, 10 минут). Если число попыток превышает заданный порог (5, 10 и т.п.), группа помечается как потенциальный bruteforce.
5. Для найденных цепочек дополнительно проверить:

- не последовала ли затем успешная аутентификация;
- не изменились ли права доступа или конфигурация системы после успешного входа.

Стандартный подход предполагает использование ausearch и ручной фильтрации. В Linux Audit Viewer этот сценарий реализован как отдельная функция: пользователь выбирает «Подбор пароля по SSH», а программа автоматически выполняет описанные шаги над уже загруженным журналом и выводит отобранные события в отдельной таблице.

2. Изменения критичных системных файлов

Цель анализа: обнаружить факты изменения критичных файлов – в первую очередь, файлов аутентификации и конфигурации служб безопасности. Такие события важны и сами по себе (подмена конфигурации), и как этап в цепочке атаки (закрепление доступа, создание новых учётных записей, отключение контроля).

На что смотреть в auditd:

- события SYSCALL и связанные с ними PATH/CWD, отражающие операции записи/создания/удаления;

- успешность системного вызова (success=yes);
- пути файлов из списка критичных (/etc/passwd, /etc/shadow, /etc/ssh/sshd_config, конфиги sudo, конфигурация auditd и др.);
- ключи правил (key=critical_conf, key=auth_files и т.п.), заданные при настройке.

Шаги анализа:

1. Ограничить временной интервал инцидента.
2. Отобрать события, соответствующие системным вызовам, изменяющим файловую систему (open с флагом записи, creat, unlink, rename и др.).
3. Отфильтровать только успешные операции над файлами из перечня критичных.
4. Для каждой операции определить:

- под какой учётной записью она выполнялась;
- какой процесс (exe, comm) инициировал изменение;
- какие ещё действия этот же процесс/пользователь выполнял рядом по времени.

В методике подчёркивается, что важно не только зафиксировать сам факт изменения, но и рассмотреть его в контексте: например, изменение sshd_config сразу после неуспешных попыток входа может говорить об укреплении защиты, а изменение после успешной компрометации – о попытке скрыть следы или ослабить контроль.

В Linux Audit Viewer сценарий «Изменения критичных файлов» реализован функцией find_critical_file_changes: она проходит по всем событиям, ищет успешные системные вызовы, затрагивающие пути из списка критичных, и выводит такие события в отдельный список, доступный для дальнейшего анализа.

3. Web-shell и запуск оболочки от сервисных пользователей

Цель анализа: выявить случаи запуска командной оболочки (или других интерпретаторов) от имени сервисных пользователей – обычно это веб-сервер

(www-data, nginx, apache и т.п.). Такой запуск нередко является индикатором эксплуатации уязвимости веб-приложения и получения интерактивного доступа к системе (web-shell).

На что смотреть в auditd:

- события SYSCALL, связанные с системным вызовом execve;
- поля exe и comm, указывающие на оболочки (/bin/bash, /bin/sh, dash) или интерпретаторы (php, python, perl и др.);
- идентификатор пользователя (UID) и имя учётной записи, относящиеся к сервисным пользователям веб-сервера;
- текущий рабочий каталог и пути к файлам (часто это каталоги веб-приложений, временные директории для загрузок и т.п.).

Шаги анализа:

1. Отобрать события execve, в которых exe/comm соответствует оболочке или интерпретатору.
2. Отфильтровать только те события, где пользователь входит в список сервисных (www-data, nginx, apache, соответствующие UID).
3. Для каждого такого запуска посмотреть:
   - из какого каталога он был инициирован;
   - какие файлы запускались/использовались;
   - не выполнялись ли сразу после запуска подозрительные команды (создание новых файлов, изменение прав, сетевые соединения).
4. Построить цепочку: HTTP-запрос (по веб-логам) → выполнение кода в контексте веб-сервера → запуск оболочки → дальнейшие действия.

В разработанном приложении сценарий «Web-shell» реализован как отдельная функция find_web_shell, которая программно повторяет описанную логическую последовательность: ищет execve оболочек, фильтрует их по сервисным пользователям и выдаёт компактный список потенциально опасных событий для детального изучения.

В сумме эти три сценария образуют ядро методики расследования в рамках курсовой работы: они показывают, как из низкоуровневых событий auditd получить осмысленные инциденты, а не просто «список строк в логе». При этом методика остаётся расширяемой: к тем же принципам можно привязать и другие типы атак – эскалацию привилегий, отключение средств защиты, массовое чтение конфиденциальных файлов и т.п., добавив соответствующие правила auditd и сценарии анализа поверх них. Набор правил, используемый для регистрации необходимых классов событий (аутентификация, изменения критичных файлов, запуск оболочек), приведён в GitHub-репозитории проекта

3 Разработка программного средства Linux Audit Viewer

3.1 Архитектурные решения и структура проекта

Разрабатываемое программное средство Linux Audit Viewer изначально проектировалось не как «монолитный скрипт для разового запуска», а как модульное приложение, ориентированное на дальнейшую доработку и расширение. В основе лежит классическая идея разделения на:

- слой получения и предварительной обработки данных (чтение и парсинг журналов auditd);
- слой доменной логики (сценарии выявления инцидентов, агрегирование статистики);
- слой представления и взаимодействия с пользователем (графический интерфейс на базе PyQt5 с табличным представлением данных).

Точка входа main.py содержит минимальный код и отвечает только за создание экземпляра QApplication, инициализацию главного окна MainWindow и запуск главного цикла обработки событий. Вся остальная логика вынесена в пакет audit_viewer, что делает структуру проекта прозрачной и удобной для описания в отчёте.

Отдельный модуль audit_helper.py вынесен за пределы пакета намеренно: он запускается с повышенными привилегиями (через pkexec) и выполняет лишь одну задачу – прочитать системный файл /var/log/audit/audit.log, распарсить его

с помощью того же механизма, что используется в основном приложении, и вернуть результат в виде JSON. Основное GUI-приложение, таким образом, всегда работает от непривилегированного пользователя и получает уже готовый список событий, не имея прямого доступа к защищённому журналу. Это снижает риски, связанные с безопасностью, и соответствует общим рекомендациям по минимизации доверенного кода.

Центральным элементом архитектуры является класс MainWindow, который наследуется от QMainWindow и классов трёх основных вкладок:

- EventsTabMixin – логика вкладки «События аудита»;
- IncidentsTabMixin – логика вкладки «Инциденты»;
- StatsTabMixin – логика вкладки «Статистика».

Такой подход позволяет: отделить интерфейс и логику каждой вкладки в отдельный модуль, не раздувая основной класс окна; использовать общее состояние (список всех событий, выбранный набор инцидентов, параметры фильтрации) через поля MainWindow; при необходимости добавлять новые вкладки без изменения уже существующего кода.

Главное окно создаёт:

- центральный виджет и компоновку;
- QTabWidget с вкладками «События аудита», «Инциденты», «Статистика», «Настройки»;
- меню «Файл» (загрузка журнала из файла, загрузка системного журнала с помощью pkexec, выход) и меню «Справка»;
- строку состояния, где выводится информация о количестве загруженных событий и служебные сообщения.

Данные, полученные либо из файла журнала, либо через audit_helper.py, помещаются в поле self.all_events. Это единый список структурированных событий аудита, с которым работают все вкладки. Для результатов сценарного анализа используется отдельное поле self.incident_events. Таким образом, MainWindow выполняет роль «центрального контроллера», который принимает

на вход «сырые» данные, превращает их в единообразную модель и передаёт эту модель во вкладки, управляющие визуализацией и прикладным анализом.

Отдельного упоминания заслуживает модуль models.py, в котором реализована таблица событий на базе стандартного механизма модель-представление Qt:

- AuditEventsTableModel наследуется от QAbstractTableModel и предоставляет табличное представление списка словарей-событий;
- PlaceholderTableView наследуется от QTableView и добавляет возможность отображать текст-заглушку при отсутствии данных (например, «Загрузите журнал через меню "Файл"»).

Использование встроенной модели Qt позволяет легко подключать сортировку по столбцам, корректно обрабатывать выбор строк и повторно применять один и тот же компонент в разных вкладках («События» и «Инциденты»), не дублируя код.

В целом архитектура проекта соответствует типичному для Qt подходу: чёткое отделение модели данных от представления, использование сигналов и слотов для реакции интерфейса на изменения данных и централизованное управление состоянием в главном окне.

3.2 Представление данных журнала auditd в программе

Чтобы методика расследования инцидентов, описанная во второй главе, могла быть реализована программно, необходимо было привести разнотипные строки журнала auditd к единой модели события, с которой могли бы работать как фильтры, так и сценарии выявления инцидентов и модуль статистики.

В рамках Linux Audit Viewer каждое событие представлено в виде словаря со стандартным набором полей, среди которых:

- timestamp – метка времени в виде числа (секунды от эпохи Unix), полученная из фрагмента audit(…:ID);
- event_type – тип записи (SYSCALL, PATH, USER_AUTH, USER_LOGIN и др.), соответствующий полю type=... в оригинальном логе;

- user – имя пользователя или идентификатор, приведённый к человекочитаемому виду (на основе UID/AUID);
- success – логическое значение, отражающее успешность операции (на основе поля success=yes/no, при наличии);
- key – значение поля key, заданное в правилах auditd (например, auth_files, web_shell, ssh_config);
- exe – путь к исполняемому файлу (exe=/usr/sbin/sshd, /bin/bash и т.п.);
- comm – командное имя процесса (comm="sshd", comm="bash" и т.п.);
- details – словарь с дополнительными параметрами события: PID, PPID, UID, GID, пути к файлам, IP-адреса, номера системных вызовов и пр.;
- raw – исходная строка (или агрегированный фрагмент) журнала, пригодная для прямого отображения пользователю.

Парсер (parser.py) отвечает за то, чтобы:

- прочитать лог строка за строкой;
- сгруппировать физические строки по идентификатору события (event ID);
- выделить ключевые поля и привести их к нормализованному виду;
- сформировать на выходе список словарей указанной структуры.

Такое представление данных напрямую «поддерживает» методику расследования:

- вкладка «События аудита» использует timestamp, event_type, user, success, key, exe, comm и часть полей details для фильтрации и отображения событий в таблице;
- вкладка «Инциденты» применяет сценарии анализа (incidents.py), которые ищут в списке событий:

– цепочки неуспешных аутентификаций по SSH (по типу события, полю success, имени процесса sshd, адресу источника);

– успешные изменения критичных файлов (по типам SYSCALL/PATH, полям success, details["path"] и ключам правил);

– запуски оболочек от сервисных пользователей (по полям exe, comm, user и идентификаторам пользователей);

- вкладка «Статистика» агрегирует события по типам (event_type), пользователям (user), датам (на основе timestamp), а также повторно использует те же сценарии для подсчёта числа изменений критичных файлов и других сводных показателей.

При этом поле raw сохраняет связь с оригинальным журналом: в любой момент аналитик может открыть «сырой» текст записи и убедиться, что интерпретация события корректна. Это важно с точки зрения доверия к инструменту: программа не скрывает от пользователя исходные данные auditd, а лишь помогает их структурировать и фильтровать.

Единая модель события также делает программу расширяемой. Чтобы добавить новый сценарий инцидента, достаточно:

1. Описать критерии отбора событий на языке полей timestamp, event_type, user, success, key, exe, comm, details и т.п.
2. Реализовать соответствующую функцию в модуле incidents.py.
3. Добавить новый пункт во вкладку «Инциденты», который будет вызывать эту функцию и показывать полученный список событий.

Таким образом, на уровне проектирования модель данных выступает связующим звеном между исходными текстовыми журналами auditd, теоретической методикой расследования инцидентов и конкретной реализацией прикладного инструмента Linux Audit Viewer.

Подробная реализация перечисленных модулей, а также вспомогательных классов и функций приведена в исходном коде проекта, размещённом в репозитории GitHub.

4 Расследование инцидентов с использованием Linux Audit Viewer

Разработанное программное средство Linux Audit Viewer служит прикладным инструментом реализации описанной ранее методики расследования инцидентов на базе подсистемы аудита Linux. Оно не заменяет сам auditd и его стандартные утилиты, а работает поверх уже сформированного журнала /var/log/audit/audit.log, предоставляя пользователю более удобный способ просмотра и анализа событий.

Практический процесс расследования с использованием программы можно условно разделить на несколько шагов.

1. Подготовка журнала для анализа

На этом этапе администратор или аналитик:

- убеждается, что подсистема auditd включена и настроены необходимые правила аудита (логирование аутентификации, изменений критичных файлов, запусков оболочек и т.п.);
- определяет временной интервал, за который требуется провести расследование (например, период, когда было зафиксировано подозрительное поведение).

Для анализа можно использовать:

- либо «живой» системный журнал /var/log/audit/audit.log, к которому доступ имеется только с правами root;
- либо заранее сохранённую копию журнала или его фрагмента (например, выгруженную с боевой системы и перенесённую на стенд).

Linux Audit Viewer поддерживает оба варианта: загрузка файла журнала в офлайн-режиме и чтение текущего системного журнала через вспомогательный скрипт с правами root.

2. Загрузка журнала в программу

Пользователь запускает Linux Audit Viewer и через меню «Файл» выбирает один из пунктов:

- «Открыть журнал из файла…» – для анализа заранее сохранённого audit.log (или его фрагмента);

- «Загрузить системный журнал (root)» – для чтения /var/log/audit/audit.log с использованием pkexec и скрипта audit_helper.py.

В обоих случаях после успешного чтения и парсинга программа формирует внутренний список событий self.all_events. Главное окно отображает:

- количество загруженных записей в строке состояния;
- доступные типы событий и пользователей в элементах фильтров;
- допустимый временной интервал (минимальная и максимальная метка времени) на вкладках «События аудита» и «Статистика».

При отсутствии или ошибке чтения журналов пользователь получает понятное сообщение (диалог с текстом ошибки либо заглушка в таблицах).

3. Первичный обзор с помощью вкладки «Статистика»

Перед тем как углубляться в детали, методически целесообразно получить общую картину активности за интересующий период. Для этого используется вкладка «Статистика»:

- задаётся временной интервал (по умолчанию – весь диапазон, покрываемый журналом);
- выполняется расчёт сводных показателей:

– общее количество событий;

– количество уникальных пользователей и типов событий;

– число неуспешных попыток аутентификации;

– количество выявленных изменений критичных файлов (по соответствующему сценарию);

- строятся визуальные распределения:

– по типам событий (какие преобладают);

– по пользователям (кто наиболее активен);

– по датам (есть ли аномальные всплески активности).

На этом шаге аналитик решает какие пользователи и временные интервалы требуют более детального рассмотрения и нет ли ярко выраженных аномалий

(например, резкого роста числа неуспешных входов или массовых изменений конфигурации).

4. Детальный анализ на вкладке «События аудита»

Далее пользователь переходит на вкладку «События аудита», где отображается табличный список событий с возможностью тонкой фильтрации. Типичный порядок действий:

1. Установить временной интервал, соответствующий интересующему периоду (например, по результатам статистики или по времени, когда был зафиксирован инцидент другими средствами).

2. Ограничить список событий по типу (например, сначала посмотреть только USER_AUTH/USER_LOGIN для анализа аутентификации, затем – SYSCALL/PATH для анализа операций с файлами).

3. При необходимости выбрать одного конкретного пользователя или оставить «Любой», если важно увидеть полную картину.

4. Задать фильтр по успешности (например, показать только события с ошибкой для расследования попыток подбора пароля).

5. Использовать поле key для отбора событий с конкретными ключами правил (auth_files, ssh_config, web_shell и т.п.).

6. При необходимости воспользоваться текстовым поиском по полям comm, exe и raw, чтобы найти выполнение конкретных программ или фрагменты команд.

Отфильтрованные события выводятся в таблицу. При выборе строки: в нижней панели «Структура» показываются все поля события в удобном виде, а так же на вкладке «Сырой лог» отображается оригинальная строка (или строки) из журнала auditd.

Это позволяет одновременно видеть и структурированную интерпретацию события, и его исходное текстовое представление, что важно для проверки корректности анализа.

5. Сценарный анализ инцидентов на вкладке «Инциденты»

Для реализации наиболее важных фрагментов методики (конкретных сценариев, как например подбор пароля по SSH, изменения критичных файлов, web-shell) программа предоставляет отдельную вкладку «Инциденты», где логика анализа уже «упакована» в сценарии.

Порядок работы здесь следующий:

1. Пользователь выбирает интересующий сценарий в списке:

- «Подбор пароля по SSH»;
- «Изменения критичных файлов»;
- «Web-shell (shell от сервисного пользователя)» и т.п.

2. Программа вызывает соответствующую функцию из модуля incidents.py, которая:

- перебирает все события в self.all_events;
- применяет критерии сценария (тип события, успешность, пользователь, адрес источника, имя процесса, путь к файлу и т.п.);
- формирует список событий, удовлетворяющих этим критериям.

3. Результаты выводятся в отдельную таблицу, аналогичную основной таблице на вкладке «События аудита», но содержащую только потенциально инцидентные записи.

4. При выборе конкретного инцидента пользователь видит его подробные детали и сырой лог, что позволяет оценить контекст (например, какие команды выполнял процесс, какие файлы затрагивались) и продолжить анализ уже на вкладке «События аудита», используя дополнительные фильтры и переходя к соседним по времени событиям.

Таким образом, вкладка «Инциденты» реализует полуавтоматический поиск интересных цепочек событий, снимая с пользователя необходимость каждый раз вручную формулировать сложные условия выборки.

6. Формирование выводов и фиксация результатов

На заключительном этапе аналитик, опираясь на найденные события и восстановленную хронологию, формулирует выводы о характере инцидента:

- был ли подбор пароля успешным или ограничился неуспешными попытками;
- какие именно критичные файлы и когда были изменены, у какими пользователями;
- имели ли место запуски оболочки от сервисных пользователей и какие действия за ними следовали.

Результаты могут затем быть оформлены в виде отдельного отчёта по инциденту, а также использованы для корректировки правил auditd и настроек самой программы (например, добавление новых ключей и сценариев).

5 Тестирование и оценка эффективности

После разработки программного средства Linux Audit Viewer была проведена проверка его работоспособности на наборе тестовых данных, сформированных с использованием реальной подсистемы аудита auditd. Цель тестирования заключалась не в формальном покрытии всех возможных ситуаций, а в практической проверке того, что реализованный функционал корректно поддерживает разработанную методику расследования инцидентов и не содержит явных ошибок, мешающих использованию программы.

Для тестирования использовался следующий подход:

- предварительно была настроена подсистема auditd с правилами, обеспечивающими регистрацию событий аутентификации, изменений критичных файлов и запусков оболочки от сервисных пользователей;
- на тестовой системе были сознательно сгенерированы типовые сценарии инцидентов (серия неуспешных попыток входа по SSH, изменение конфигурационного файла, запуск оболочки от имени сервисного пользователя), а также обычная повседневная активность, чтобы проверить поведение программы на «смешанном» журнале;

- журнал auditd за соответствующий период был сохранён и использован в качестве основного тестового набора.

В ходе тестирования последовательно проверялись следующие группы функций.

1. Загрузка и разбор журналов аудита

Проверялась корректность работы с двумя вариантами источника данных:

- загрузка журнала из заранее сохранённого файла через пункт меню «Открыть журнал из файла…»;
- загрузка актуального системного журнала /var/log/audit/audit.log с использованием вспомогательного скрипта audit_helper.py, запускаемого через pkexec.

В обоих сценариях подтверждено, что:

- при корректном журнале программа успешно загружает данные, формирует внутренний список событий и отображает в строке состояния фактическое количество записей;
- элементы фильтрации (типы событий, пользователи, диапазон времени) и вкладка статистики автоматически подстраиваются под загруженный набор;
- при попытке открытия несуществующего или некорректного файла приложение выдаёт понятное сообщение об ошибке и не переходит в неконсистентное состояние (таблицы либо остаются пустыми с текстом-заглушкой, либо отображают ранее загруженные данные без сбоев).

2. Работа вкладки «События аудита» и механизмов фильтрации

На следующем этапе тестировалась основная вкладка «События аудита», обеспечивающая просмотр и фильтрацию событий. На основании заранее известных событий в тестовом журнале было проверено:

- фильтрация по времени: при установке узкого временного интервала в таблице остаются только события, реально приходящиеся на указанный диапазон;

- фильтрация по типу события: при выборе, например, USER_AUTH в таблице отображаются только события аутентификации, а при выборе SYSCALL – только системные вызовы;
- фильтрация по пользователю: при выборе конкретной учётной записи в таблице остаются только события, связанные с данным пользователем;
- фильтрация по признаку успешности: режим «только с ошибкой» корректно позволяет выделять неуспешные попытки аутентификации и другие операции с success=no;
- фильтрация по ключу правила (key): при задании, например, auth_files или ssh_config в поле фильтра отображаются только события, сгенерированные соответствующими правилами auditd;
- текстовый поиск: поиск по фрагментам команд (bash, sshd) и путям (/etc/ssh/sshd_config) корректно отбирает события, где соответствующие значения встречаются в полях comm, exe или в исходной строке журнала.

Отдельно было проверено, что при выборе строки в таблице:

- в панели «Структура» отображаются все ключевые поля события (тип, пользователь, путь к исполняемому файлу, параметры системного вызова и т.п.);
- вкладка «Сырой лог» содержит исходный текст записи auditd, соответствующий тому, что хранится в тестовом журнале.

Таким образом, подтверждено, что вкладка «События аудита» позволяет воспроизводимо выполнять описанные в методике шаги фильтрации и детального изучения отдельных событий.

3. Проверка сценариев выявления инцидентов

Особое внимание при тестировании уделялось вкладке «Инциденты», так как она непосредственно реализует ключевые сценарии методики расследования. Для каждого сценария были заранее сгенерированы соответствующие события, после чего проверялось:

- сценарий «Подбор пароля по SSH» корректно выделяет последовательность неуспешных попыток входа по SSH от одного источника и

к одному пользователю в пределах заданного временного окна; в результирующем списке присутствуют все тестовые попытки, а выбор случайных «нормальных» событий без серии ошибок не наблюдается;

- сценарий «Изменения критичных файлов» выявляет события успешного изменения заранее определённых критичных файлов (в том числе тестового изменения конфигурации SSH), при этом в деталях события корректно отображаются пользователь, инициировавший изменение, и путь к изменённому файлу;
- сценарий «Web-shell» находит запуск оболочки (/bin/bash или аналогичной) от имени сервисного пользователя, использованного в тесте, и не включает в результаты обычные интерактивные оболочки, запускаемые обычными пользователями.

В каждом случае дополнительно визуально сравнивалось содержимое найденных событий в программе с соответствующими строками в исходном журнале auditd, что подтвердило корректность парсинга и работы критериев отбора.

4. Проверка статистического блока

Для вкладки «Статистика» проводилась сверка рассчитанных программой величин с ожидаемыми значениями:

- общее количество событий в выбранном временном интервале совпало с числом записей, отображаемых на вкладке «События аудита» при тех же границах интервала;
- количество уникальных пользователей и типов событий соответствовало фактическому содержимому журнала;
- число неуспешных попыток аутентификации совпало с результатами выборки по USER_AUTH/USER_LOGIN с success=no;
- количество изменений критичных файлов соответствовало числу событий, выдаваемых сценарием «Изменения критичных файлов» на вкладке «Инциденты».

Также проверялась корректность обновления графиков и таблиц при изменении временного интервала: при сужении окна статистики гистограммы и линии на графиках изменялись ожидаемым образом, что дополнительно подтверждает правильность отбора событий по времени.

5. Общие выводы по результатам тестирования

Проведённые проверки показали, что:

- программа корректно загружает и разбирает журналы auditd как из файлов, так и из системного журнала с использованием вспомогательного скрипта;
- механизмы фильтрации и просмотра деталей событий на вкладке «События аудита» позволяют воспроизводимо выполнять основные шаги методики расследования;
- реализованные сценарии выявления инцидентов (подбор пароля по SSH, изменения критичных файлов, запуск оболочки от сервисных пользователей) работоспособны на практических тестовых примерах и корректно выделяют соответствующие события;
- статистический блок адекватно отражает структуру событий в журнале и согласован с результатами фильтрации и сценарного анализа.

В совокупности это позволяет сделать вывод, что разработанное программное средство в текущей версии работоспособно и пригодно для использования в рамках предложенной методики расследования инцидентов. При этом возможные направления дальнейшего развития (расширение набора сценариев, интеграция с базой данных, онлайн-мониторинг) относятся уже к перспективам доработки и не влияют на общую оценку корректности базового функционала, проверенного в ходе тестирования.

6 Практическая значимость и перспективы развития

Разработанная в работе методика расследования инцидентов на базе подсистемы аудита Linux (auditd) и реализованное программное средство Linux

Audit Viewer имеют практическую ценность как для учебных целей, так и для реального применения на тестовых и малонагруженных продуктивных системах.

Во-первых, методика опирается на стандартный механизм аудита, доступный во всех современных Linux-дистрибутивах, и не требует установки специализированных коммерческих средств. Для включения методики в практику достаточно:

- настроить базовый набор правил auditd, обеспечивающих регистрацию событий аутентификации, изменений критичных файлов и запусков оболочки от сервисных пользователей;
- организовать регулярный сбор журналов /var/log/audit/audit.log;
- использовать Linux Audit Viewer как инструмент для первичного анализа и расследования инцидентов.

Это делает методику удобной для учебных лабораторных работ и демонстраций: студент получает возможность увидеть полный цикл от настройки аудита до анализа реальных событий, не выходя за рамки стандартных компонентов Linux. Программа выступает при этом в роли «моста» между формальным логом и человеческим восприятием: вместо громоздких строк журнала пользователь работает с таблицами, фильтрами, сценариями и визуальной статистикой, что снижает порог входа и позволяет сосредоточиться на сути расследования.

Во-вторых, Linux Audit Viewer может использоваться как вспомогательный инструмент администратором или специалистом по безопасности на небольших серверах, где развертывание полноценных SIEM-систем или централизованных платформ логирования нецелесообразно. В таких условиях типовая задача выглядит как «есть файл audit.log, нужно быстро понять, что происходило в системе в момент инцидента» – и именно под такую задачу программа и методика были ориентированы. Возможность анализа заранее сохранённого журнала позволяет использовать инструмент и для постфактум-разборов, и для разборов на стендовых копиях систем.

Отдельная практическая значимость заключается в том, что:

- реализованные сценарии (SSH-bruteforce, изменения критичных файлов, web-shell) демонстрируют, как из низкоуровневых логов auditd выделять осмысленные инциденты и строить хронологию действий нарушителя;
- единая модель данных события и вкладка статистики задают задел для дальнейшей автоматизации анализа (например, для более сложных правил корреляции и поиска аномалий);
- использование вспомогательного скрипта для доступа к системному журналу с правами root показывает корректный подход к разделению привилегированного и пользовательского кода.

При этом текущая реализация не претендует на полноту и промышленный уровень, а рассматривается как базовая платформа, вокруг которой могут развиваться более сложные решения. Возможные направления дальнейшего развития включают:

- Расширение набора сценариев инцидентов.

На основе уже существующей модели событий можно реализовать дополнительные сценарии: эскалация привилегий (по сочетаниям setuid-вызовов и изменения групп), отключение или изменение настроек auditd, массовое чтение конфиденциальных файлов, нетипичная активность определённых учетных записей и т.п. Добавление новых функций в модуль incidents.py и соответствующих элементов интерфейса органично вписывается в существующую архитектуру.

- Интеграция с базой данных.

В текущей версии все события загружаются в оперативную память и анализируются в рамках одной сессии. Для больших журналов или длительных расследований может быть полезна выгрузка событий в СУБД (например, SQLite или PostgreSQL) с последующей фильтрацией на уровне SQL-запросов и возможностью сохранять результаты расследований. Архитектура программы уже предполагает наличие единой модели данных, которую можно «подменить» реализацией поверх базы без изменения пользовательского интерфейса.

- Режим онлайн-мониторинга.

Сейчас программа ориентирована на анализ уже сформированного журнала. Перспективным направлением является добавление режима «живого» мониторинга, при котором новые события из audit.log (или из сокета auditd) отображаются в интерфейсе по мере поступления. Это позволит использовать Linux Audit Viewer не только для постфактум-расследований, но и как простое средство оперативного мониторинга активности в системе.

- Экспорт отчётов и артефактов расследования.

Для формализации результатов расследования может быть полезен экспорт найденных событий и сводной статистики в удобные форматы – CSV, HTML, PDF. Это позволит прикладывать к отчётам по инцидентам не только словесное описание, но и конкретные выборки из журнала аудита, сформированные программой, что повышает наглядность и повторяемость анализа.

- Глубже интеграция с другими средствами безопасности.

В дальнейшем Linux Audit Viewer может рассматриваться как компонент более широкой системы: например, получать от внешнего источника сигналы о подозрительной активности (по времени или пользователю) и автоматически подготавливать соответствующие выборки из журналов auditd. Возможен и обратный вариант – экспорт найденных инцидентных событий в сторонние системы для последующей корреляции.

С точки зрения учебного процесса выполненная работа демонстрирует полный цикл: от анализа подсистемы аудита и формулировки методики расследования до реализации и тестирования прикладного инструмента. Это подчёркивает, что проект имеет не только теоретическую ценность, но и практическую направленность: методика может быть применена на любом стандартном Linux-хосте, а программа реально упрощает работу с журналами аудита по сравнению с ручным разбором текстовых файлов и использованием только консольных утилит.

ЗАКЛЮЧЕНИЕ

В ходе курсовой работы была достигнута поставленная цель – на практике исследованы возможности подсистемы аудита Linux (auditd) для анализа событий информационной безопасности и показано, как на их основе строится методика расследования инцидентов, поддержанная прикладной программой.

Сначала были разобраны роль системного аудита в расследовании и особенности auditd: архитектура, формат журналов, типы событий и ключевые поля. Показано, что низкоуровневые события auditd позволяют восстанавливать действия пользователей и процессов с нужной детализацией для последующего анализа.

Затем сформирована методика расследования инцидентов с использованием auditd, включающая этапы подготовки (настройка правил аудита), фиксации инцидента, извлечения и фильтрации журнала, детального анализа и формулирования выводов. Отдельно описаны три типовых сценария: подбор пароля по SSH, изменения критичных файлов и запуск оболочки от сервисных пользователей.

На базе этой методики разработано графическое приложение Linux Audit Viewer, работающие со стандартным журналом /var/log/audit/audit.log. Программа обеспечивает удобный просмотр и фильтрацию событий, сценарный поиск инцидентов и обзорную статистику, что существенно упрощает практическое применение методики по сравнению с ручным анализом текстовых логов.

Тестирование показало корректную работу загрузки и разбора журналов, фильтров, сценариев выявления инцидентов и статистического блока. Намечены перспективы развития, но текущая версия уже может использоваться как учебный и прикладной инструмент для анализа журналов аудита в Linux.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Ефанов, Д. В. Базовые механизмы защиты ядра Linux : учебное пособие / Д. В. Ефанов. – Москва : НИЯУ МИФИ, 2022. – 192 с. – ISBN 978-5-7262-2924-9. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/355520 (дата обращения: 03.12.2025). – Режим доступа: для авториз. пользователей.

2 Red Hat Enterprise Linux 8. Security hardening. Chapter 11. Auditing the system [Электронный ресурс]. // Red Hat, Inc. – Режим доступа: URL: https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/8/html/security_hardening/auditing-the-system_security-hardening (дата обращения: 03.12.2025).

3 Boelen M. Linux Audit Framework 101 – Basic Rules for Configuration [Электронный ресурс] // Linux Audit. – 2025. – Режим доступа: URL: https://linux-audit.com/linux-audit-framework/linux-audit-framework-101-basic-rules-for-configuration/ (дата обращения: 03.12.2025).

4 Roth F. Neo23x0/auditd: Best Practice Auditd Configuration [Электронный ресурс]. // GitHub, Inc. – Режим доступа: URL: https://github.com/Neo23x0/auditd (дата обращения: 03.12.2025).

5 Linux Audit Viewer [Электронный ресурс] – репозиторий программного проекта – Режим доступа: URL: https://github.com/uTakCouDeT/Linux-Audit-Viewer (дата обращения: 03.12.2025).