ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «УРАЛЬСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ ИМЕНИ ПЕРВОГО ПРЕЗИДЕНТА РОССИИ Б.Н. ЕЛЬЦИНА»

Институт радиоэлектроники и информационных технологий-РтФ

Учебно-научный центр «Информационная безопасность»

На правах рукописи

Гибилинда Роман Владимирович

# РАЗРАБОТКА АВТОМАТИЗИРОВАННЫХ МЕТОДОВ АНАЛИЗА ВОЗДЕЙСТВИЙ НА ФАЙЛЫ В ЗАДАЧЕ РАССЛЕДОВАНИЯ ИНЦИДЕНТОВ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Специальность 2.3.6 – Методы и системы защиты информации, информационная безопасность

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:
кандидат технических наук, доцент,
Синадский Николай Игоревич

Екатеринбург – 2021

# ОГЛАВЛЕНИЕ

# Введение

**Актуальность темы исследования**. Вступление в силу Федерального закона № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации» [1] и других сопутствующих нормативных правовых актов обязало владельцев информационных систем (ИС), входящих в состав критической информационной инфраструктуры (КИИ), принимать меры по защите информации, в частности проводить аудит информационной безопасности (ИБ) ИС и расследование инцидентов.

Под аудитом ИБ, согласно [2], будем понимать «систематический, независимый и документируемый процесс получения свидетельств деятельности организации по обеспечению информационной безопасности и установлению степени выполнения в организации критериев информационной безопасности». Одной из задач аудита [3] является выявление уязвимостей в ИС, в т.ч. эксплуатируемых как локально, так и удаленно, которые могут быть использованы злоумышленником для нарушения действующей политики безопасности (осуществление несанкционированного доступа, нарушение конфиденциальности, целостности и доступности обрабатываемых в ИС сведений и др.) с целью формирования рекомендаций по совершенствованию мер и методов обеспечения ИБ, направленных на устранение выявленных уязвимостей.

Совершенствование существующих и создание новых методов защиты информации является актуальной задачей, которая нашла свое отражение в трудах ученых и практиков [4-143]. Несмотря на существование множества решений, предназначенных для обеспечения требуемого (согласно принятой в организации политики безопасности) уровня безопасности информации, злоумышленники не снижают своей активности по реализации различных видов компьютерных атак (КА), направленных на получение несанкционированного доступа к ИС и нарушение конфиденциальности, целостности и доступности обрабатываемых в ИС сведений.

Для определения действий злоумышленника и последствий возникшего инцидента ИБ проводится расследование инцидента. Результаты расследования

могут указывать на не выявленные недостатки системы обеспечения ИБ и действующей политики безопасности организации и составляют основу рекомендаций по совершенствованию мер по защите информации, направленных на недопущение возникновения подобных инцидентов ИБ в будущем.

В процессе расследования инцидента ИБ возникает потребность в определении воздействий на информацию, обрабатываемую в ИС. Хранилищем информации в ИС является файл. Таким образом, для определения воздействий на информацию следует обнаружить и классифицировать воздействия на файлы, предварительно их идентифицировав.

Основой для определения воздействий на файлы является информация, содержащаяся в разноформатных массивах данных (рисунок 1.1).

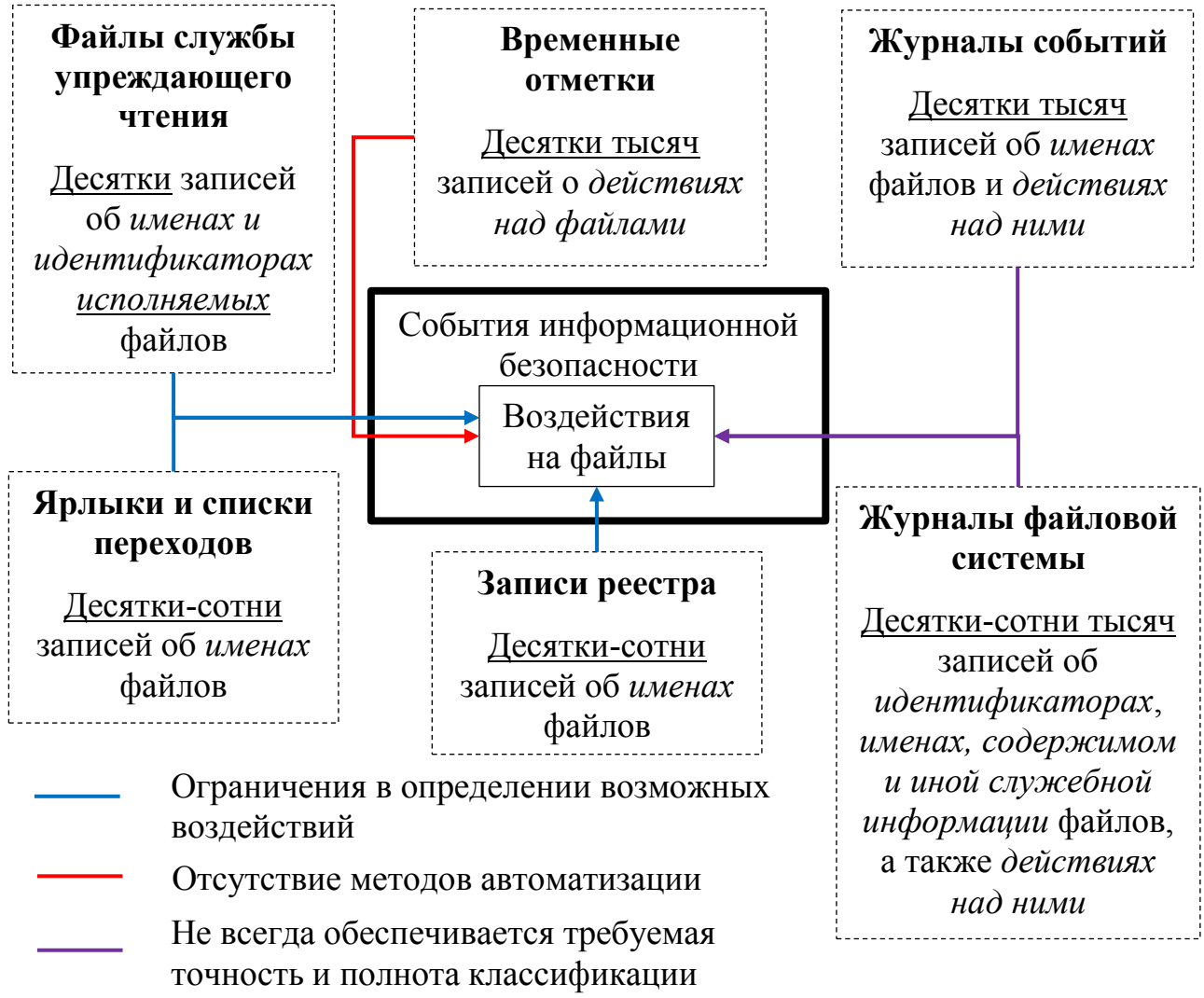


Рисунок 1.1. Массивы данных и недостатки методов их анализа

Как видно из рисунка 1.1, в процессе анализа информации о воздействиях на файлы, содержащейся в разноформатных массивах данных, специалист, проводящий расследование инцидентов ИБ, сталкивается с недостатками существующих методов анализа данных (рассмотрены подробно в первой главе диссертации): отсутствие методов автоматизации анализа или неудовлетворительные результаты классификации с позиций точности и полноты, а также ограничения в полноте определяемых воздействий.

Таким образом, для решения задачи автоматизации процесса обнаружения и классификации воздействий на файлы в рамках расследования инцидентов ИБ использование существующих методов анализа данных либо не обеспечивает необходимой точности и/или полноты, либо имеет ограничения по набору определяемых воздействий. Возникает потребность в разработке новых методов анализа данных, учитывающих порядок изменения признаков, характеризующих файлы, в рамках осуществленных воздействий. Для разрешения возникшей потребности необходимо выбрать массив данных, обладающий наиболее полной информацией о воздействиях на файлы, создать алгоритмический и методологический аппарат автоматизированного анализа воздействий, менее подверженный указанным ранее недостаткам существующих методов, и на его основе разработать специальный комплекс программных средств, позволяющий ускорить процесс расследования инцидентов ИБ.

**Степень разработанности темы исследования**. Расследование инцидентов ИБ представляет собой сложный процесс, требующий наличия у специалиста обширных знаний об особенностях функционирования ИС, способах обеспечения ИБ и оценки качества принятых мер защиты, математических методах анализа данных, связанных с инцидентами.

Вопросы, связанные с разработкой и применением мер, направленных на обеспечение ИБ, рассмотрены в работах С.С. Титова [4, 5], А.А. Захарова [6], А.Н. Соколова [7], В.В. Богданова [8], Ю.Д. Королькова [9], П.Н. Девянина [10, 11], И.И. Баранковой [12]. Авторами были предложены различные методы обеспечения ИБ, как в рамках отдельного узла, так и системы в составе

вычислительной сети, описаны организационные меры, направленные на нейтрализацию угроз ИБ, рассмотрены методы и алгоритмы выявления угроз в сетевом трафике.

Применению математических методов и алгоритмов при анализе данных (в том числе используемых в рамках расследования инцидентов ИБ) посвящено множество научных работ, среди которых необходимо отметить труды В.А. Баранского [13, 14], Н.А. Гайдамакина [15], А.А. Захарова [16, 17], С.В. Поршнева [18], Д. Феррейры (D.R. Ferreira) [19], В. Сатиша (V. Sathish) [20], Е. Чуа (E. Chuah) [21], В.Н. Зуева [22], А.Н. Соколова [23, 24], О.И. Шелухина [25, 26], Ф. Юана (F. Yuan) [27], С. Йена (S. Yen) [28], К. Берлина (K. Berlin) [29], Р. Вааранди (R. Vaarandi) [30-32], Б. Штейна (B. Stein) [33], Х. Штудиавана (H. Studiawan) [34, 35]. Авторы разработали методы анализа информации, в том числе основанные на теории графов, рассмотрели применение метрик в контексте обработки информации, связанной с событиями ИБ, предложили новые и адаптировали существующие методы и алгоритмы машинного обучения для анализа разноформатных массивов данных: сетевого трафика, журналов событий и др. Часть указанных работ содержат количественные оценки результатов работы алгоритмов, что позволяет выбрать наилучший для решения задач анализа данных.

**Целью** исследования является разработка научно-обоснованных автоматизированных методов расследования инцидентов ИБ, их программных реализаций и методики использования.

Для достижения поставленной цели сформулированы и решены следующие **задачи**:

1. Анализ состояния предметной области и инструментов для автоматизированной идентификации, обнаружения и классификации воздействий на файлы.

2. Разработка математического аппарата для анализа воздействий на файлы, в частности: математической модели процесса идентификации воздействий на файлы; кластеризационного метода идентификации воздействий; метода экспресс-анализа событий ИБ.

3. Создание комплекса программных средств, реализующих математические методы для анализа воздействий на файлы, направленных, в том числе, на нарушение действующей политики безопасности.

**Объект исследования** – процесс расследования инцидентов ИБ.

**Предмет исследования** – автоматизированные методы и алгоритмы расследования инцидентов ИБ.

**Научная новизна работы**. В рамках проведенного исследования получены следующие новые научные результаты:

1. Разработана модель процесса идентификации воздействий на файлы, основанная на математическом аппарате сетей Петри, позволяющая формализовать набор признаков, характеризующих файл, для их последующего анализа в рамках расследования инцидентов ИБ (соответствует п. 14 паспорта специальности в части создания событийной модели процесса идентификации воздействий на файлы, в т.ч. направленных на нарушение действующей политики ИБ, применяемой при мониторинге состояния объекта, находящегося под воздействием угроз нарушения его ИБ).

2. Разработан кластеризационный метод идентификации воздействий на файлы, направленных, в том числе, на нарушение действующей политики ИБ (соответствует п. 15 паспорта специальности в части разработки кластеризационного метода идентификации воздействий на файлы с целью создания шаблонов воздействий как нового элемента контроля за влиянием на информацию в рамках процесса управления событиями ИБ).

3. Разработан метод экспресс-анализа событий ИБ, связанных с воздействиями на файлы, позволяющий ускорить процесс обнаружения и классификации воздействий (соответствует п. 13 паспорта специальности в части разработки принципа по созданию нового метода определения нормальных, аномальных и условно аномальных событий ИБ в целях формирования рекомендаций по совершенствованию мер, направленных на обеспечение ИБ).

**Теоретическая значимость работы** выражается в развитии научно-методического аппарата для анализа воздействий на файлы в рамках расследования инцидентов ИБ [144-151].

**Практическая значимость работы** заключается в разработке комплекса программных средств, обеспечивающего автоматизацию процесса анализа воздействий на файлы, в том числе направленных на нарушение действующей политики ИБ [148, 149].

**Методология и методы исследования**. В диссертации представлены результаты исследований, полученные с помощью математического аппарата сетей Петри, теории вероятностей, кластерного анализа и алгоритмов классификации.

**Положения, выносимые на защиту**:

1. Процесс идентификации воздействий на файлы, направленных, в том числе, на нарушение действующей политики ИБ, описывается математическим аппаратом на основе сетей Петри [146].

2. Предложенный кластеризационный метод, в котором используются адаптированные к структуре массивов данных алгоритмы подготовки входной информации и определения оптимального количества кластеров, позволяет автоматизировать процесс идентификации воздействий на файлы [147].

3. Метод экспресс-анализа событий информационной безопасности, связанных с воздействиями на файлы, и реализующий его комплекс программных средств, позволяют автоматизировать процесс выявления нормальных, аномальных и условно аномальных воздействий на файлы [144, 148, 149].

**Границы исследования** – операционная система (ОС) Windows с файловой системой (ФС) NTFS и журналом изменений тома $UsnJrnl, являющимся массивом данных о воздействиях на файлы.

**Достоверность и обоснованность** полученных результатов подтверждается адекватным выбором математического аппарата задачам исследования и результатами экспериментальной апробации предложенных моделей и методов анализа воздействий на файлы.

**Апробация исследования.** Основные результаты диссертационных исследований докладывались на различных семинарах и совещаниях, а также на международной научной конференции, в том числе:

1. II Всероссийская научная конференция (с приглашением зарубежных ученых) «Фундаментальные проблемы информационной безопасности в условиях цифровой трансформации» (FISP-2020), 30 ноября 2020 г., Россия, г. Ставрополь [150];

2. 16-я Юбилейная международная молодежная научно-техническая конференция «Современные проблемы радиоэлектроники и телекоммуникаций, РТ-2020», 12-16 октября 2020 г., Россия, г. Севастополь [151];

3. 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), 14-15 мая 2020 года, Россия, г. Екатеринбург [145].

**Внедрение результатов исследования.** Результаты работы используются в ООО «Уральский центр систем безопасности», Екатеринбург, Россия (акт об использовании результатов от 16.03.2021); в Уральском федеральном университете имени первого Президента России Б.Н. Ельцина, Екатеринбург, Россия (акт об использовании результатов от 10.03.2021); в ОКБ «Новатор», Екатеринбург, Россия (акт об использовании результатов от 23.12.2020).

**Публикации**. По теме диссертации опубликовано 6 научных работ, из них 4 статьи опубликованы в рецензируемых научных журналах и изданиях, определенных ВАК РФ и Аттестационным советом УрФУ, включая 1 статью в издании, индексируемом в международной цитатно-аналитической базе Scopus. Имеются 2 свидетельства о государственной регистрации программы для ЭВМ.

**Личный вклад автора**. Все результаты и положения, выносимые на защиту, получены лично автором. Все алгоритмы, обсуждаемые в работе, разработаны и экспериментально исследованы автором самостоятельно. Научный руководитель принимал участие в постановке цели и задач исследования, их предварительном анализе, планировании экспериментов. Подготовка к публикации полученных

результатов проводилась совместно с соавторами, при этом вклад диссертанта был определяющим.

**Структура и объем работы**. Диссертация состоит из введения, трех глав, заключения и приложений. Общий объем диссертации составляет 178 страниц, включая 31 рисунок и 23 таблицы. Список использованных источников содержит 151 наименование.

# Глава 1. Анализ состояния предметной области. Постановка задач исследования

## 1.1. Понятие расследования инцидента информационной безопасности

Согласно [36], менеджмент инцидентов ИБ состоит из 4 этапов:

- планирования и подготовки;
- использования;
- анализа;
- улучшения.

На этапе планирования и подготовки составляются регламенты, разрабатываются инструкции, политика безопасности, формируется база знаний о причинах возникновения инцидентов и пр. Этап использования предназначен для реагирования на инцидент, включая правовую оценку. В рамках работы под *расследованием инцидента ИБ* будем понимать реагирование на него, но без осуществления правовой оценки. Целью этапа анализа является проработка мер, направленных на совершенствование ИБ, по результатам, полученным на этапе использования. Этап улучшения необходим для непосредственной реализации этих мер.

Причины возникновения инцидентов ИБ носят как случайный (стихийное бедствие, физический износ оборудования, приводящий к сбою), так и преднамеренный характер. В рамках работы рассматриваются те инциденты, причины которых являются преднамеренными, в частности, КА и деструктивные действия пользователя или ПО.

В ОС Windows с ФС NTFS фиксируется различная информация, связанная с действиями пользователей, которая обеспечивает как удобство работы (последние открытые файлы, служба упреждающего чтения), так и возможность контроля конфиденциальности, целостности и доступности обрабатываемой информации (журналы событий, временные отметки, записи реестра и пр.). Использование такой информации позволяет, в частности:

- отследить активность пользователя или ПО;

- определить факты запуска и возможные последствия работы вредоносного ПО (ВПО), например WannaCry [37], Jigsaw [38] и т.д.;
- выявить осуществление КА на Web-сервер, например, несанкционированную загрузку файлов.

Согласно [39], при расследовании инцидентов ИБ осуществляется процедура сбора и обработки данных, в которых может содержаться информация о возникшем инциденте ИБ – в том числе информация, в которой зафиксированы действия пользователя или ПО.

Таким образом, данные, позволяющие определить факт инцидента ИБ, его этапы и последствия, имеются в значительном количестве, но они представлены в разноформатных массивах, что затрудняет применение автоматизированных средств анализа этих данных. В то же время, объем анализируемых данных может исчисляться сотнями тысяч записей / полей / иных различных информационных структур, что затрудняет их ручной анализ. Также существует проблема искажения / уничтожения данных.

Итоговый объем обрабатываемых данных при расследовании инцидента ИБ предполагает проведение анализа с использованием средств автоматизации, анализирующих информацию, представленную в формализованном виде. Временные затраты на проведение расследования являются важным фактором, который следует учитывать при внедрении автоматизации. Одним из способов сокращения времени на обработку данных, предложенный в данной работе, является отказ от анализа активности штатного ПО ОС Windows и нормальных, санкционированных действий пользователей за счет определения совокупности значений (свойств), характерных для аномальных, несанкционированных действий пользователя или активности ВПО.

### 1.2. Понятие воздействия на файл

Причиной возникновения инцидента является событие ИБ, которое согласно [36], определяется как «идентифицированное появление определенного состояния системы, сервиса или сети, указывающего на возможное нарушение политики ИБ или отказ защитных мер, или возникновение неизвестной ранее

ситуации, которая может иметь отношение к безопасности». В соответствии с [39] инцидент ИБ определяется как совокупность событий ИБ. Опишем инцидент ИБ *A* выражением:

$$A = \left\{ \left\langle S_1,...,S_p \right\rangle \right\}, \quad (1)$$

где $S_1,...,S_p$ – события ИБ, *p* – их количество.

В рамках исследования каждое событие ИБ описывается как совокупность воздействий на файлы и описывается кортежем:

$$S_k = \left\langle \left\langle Y_{fi}^k \right\rangle_{i=1}^{l} \right\rangle_{f=1}^{n}, \quad (2)$$

где $Y_{fi}^k$ – воздействие *i* на файл *f,* относящееся к событию ИБ *k*, *l* – количество воздействий, *n* – количество файлов.

Событие ИБ может сопровождаться широким спектром как простых[1], так и сложных комплексных[2] воздействий на файлы. В основе каждого воздействия на файл лежат признаки, характеризующие файл, и соответствующая файловая операция. Таким образом, расследование инцидентов ИБ может быть рассмотрено как процесс идентификации, обнаружения и классификации воздействий на файлы, составляющих событие ИБ. Под идентификацией воздействий на файлы будем понимать процесс, в результате которого определяется порядок изменения признаков, характеризующих файл. Под обнаружением воздействия на файл будем понимать факт нахождения признаков воздействия в массиве данных КС, где произошел инцидент. Классификацией в рамках работы является процесс отнесения обнаруженного воздействия к классу аномальных или нормальных для определения хода инцидента ИБ.

В рамках настоящего исследования будем использовать понятийный аппарат, детализирующий существующие определения инцидента и события ИБ в контексте решаемой задачи — расследования инцидента ИБ.

---

[1] Простым является воздействие на файл, состоящее из одной файловой операции.

[2] Сложным комплексным называется такое воздействие, которое состоит из нескольких простых и может иметь отношение к нескольким файлам.

### 1.2.1. Признаки, характеризующие файл

Информация, обрабатываемая с помощью информационных технологий (ИТ), хранится в виде файлов. Процесс ее обработки непосредственно связан с осуществлением воздействий на файлы.

Будем считать, что произвольный файл $f$ в каждый момент времени $t$ характеризуется следующими признаками:

- $I_f(t)$ – идентификатор файла – уникальное числовое значение, содержащееся в служебной информации о файле, используемое драйвером ФС для однозначного определения файла;
- $D_f(t)$ – идентификатор родительского каталога файла – уникальное числовое значение, используемое драйвером ФС для установления однозначного соответствия между файлом и каталогом, в котором файл расположен;
- $N_f(t)$ – имя файла – битовая строка, используемая драйвером ФС для представления файла пользователю;
- $C_f(t)$ – содержимое файла – битовая строка, являющаяся информацией, хранимой в файле;
- $X_f(t)$ – иная служебная информация о файле – набор числовых значений, являющихся служебной информацией о файле, зависящий от типа ФС.

Признаки, характеризующие некоторый файл *f*, образуют вектор $V_f(t)$, описываемый выражением:

$$V_f(t) = \{I_f(t), D_f(t), N_f(t), C_f(t), X_f(t)\}. \quad (3)$$

Запишем и будем в дальнейшем использовать выражение, описывающее вектор признаков, характеризующих файл, в удобной для восприятия форме:

$$V_f(t) = V_f = \{I_f, D_f, N_f, C_f, X_f\}. \quad (4)$$

При рассмотрении некоторого файла $f$ в рамках расследования инцидента ИБ возможно определение значений компонентов вектора $V_f$, которые описывают состояние файла на момент события ИБ. Зададим определение состояния файла.

Определение 1. *Состояние файла – значения компонентов вектора характеризующих признаков $V_f$, присущих файлу в определенный момент времени t.*

### 1.2.2. Файловая операция

Изменение значений компонентов вектора $V_f$ происходит не спонтанно, а в соответствии с процессом, называемым файловой операцией. В дальнейшем будем обозначать файловую операцию символом *O*.

Пусть в момент времени $t_1$ начат процесс, в результате которого произошло изменение значений компонентов вектора $V_f$, и к моменту времени окончания процесса $t_2$ файл, описываемый начальным состоянием $V_{f1} = V_f(t_1)$, перешел в конечное состояние $V_{f2} = V_f(t_2)$, причем $V_{f1} \neq V_{f2}$.

Зададим определение файловой операции.

Определение 2. *Файловая операция O – процесс модификации значений компонентов вектора $V_f$ признаков, характеризующих файл, приводящий к переходу файла от одного состояния к другому.*

Примерами ФО являются: создание, удаление, переименование, изменение содержимого, изменение служебной информации, перемещение, копирование. Процесс изменения признаков, характеризующих файл, будет рассмотрен во второй главе настоящей работы.

### 1.2.3. Воздействие на файл

При осуществлении воздействий на файлы возможно проведение как одной, так и нескольких файловых операций, которые приведут к появлению нескольких состояний файла, являющихся последовательным переходом от состояния до начала воздействия к окончательному состоянию. Также определены некоторые типы воздействий на файлы, осуществление которых приводит к осуществлению набора файловых операций с несколькими файлами.

Определение 3. *Воздействие на файл – совокупность файловых операций, связанных по назначению, разделенных по времени и приводящих к изменению состояний одного или нескольких файлов.*

Следует отметить, что в зависимости от типа воздействия могут возникать ситуации, когда воздействие на один файл приводит к осуществлению файловых операций над другими файлами, но в рамках текущего воздействия. Такие ситуации характерны для сложных комплексных воздействий. Ярким примером является работа с временными файлами в процессе редактирования, например документа в формате docx.

### 1.3. Анализ массивов данных в ОС Windows при расследовании инцидентов ИБ

Расследование инцидентов ИБ связано с анализом множества разноформатных массивов данных в ОС Windows, присущих компьютерной системе (КС) и содержащих информацию о воздействиях на файлы (далее по тексту — массивы данных):

- временные отметки файлов;
- журналы событий ОС Windows;
- ярлыки последних открытых файлов и списки переходов;
- файлы службы упреждающего чтения Superfetch;
- записи реестра;
- журналы файловой системы NTFS.

Список не является исчерпывающим, но состоит из присутствующих по-умолчанию массивов данных. Рассмотрим достоинства и недостатки каждого из указанных массивов данных при расследовании инцидента ИБ.

Временной отметкой файла в ФС NTFS называется числовое значение, располагающееся в служебной информации $X_f$ о файле. Несколько ВО, имеющих отношение к файлу, назовем комбинацией ВО. При рассмотрении нескольких файлов в рамках расследования инцидента ИБ необходимо определить комбинации ВО для каждого файла. Вместе эти комбинации являются массивом данных, содержащим информацию о воздействиях на файлы.

Размерность ВО в ФС NTFS равна 8 байтам, точность – 100 нс. Значение ВО фиксирует количество 100 нс интервалов, прошедших с 00:00 01.01.1601 года

[40]. Служебная информация о файле $X_f$ в ФС NTFS содержит 8 ВО в двух атрибутах – $STANDARD_INFORMATION и $FILE_NAME по 4 в каждом: ВО создания файла, ВО модификации содержимого файла $C_f$, ВО модификации служебной информации о файле $X_f$, ВО последнего доступа к файлу [40].

На основании значений ВО в каждом из указанных атрибутов возможно определение осуществленной файловой операции. Например: при создании файла ВО в $STANDARD_INFORMATION и $FILE_NAME устанавливаются равными на момент совершения файловой операции, а при перемещении файла в пределах тома изменяется ВО последнего доступа и модификации служебной информации о файле $X_f$ [40].

Chow K., Law F., Kwan M., Lai K. в работе [41] рассматривают лишь 3 ВО из атрибута $STANDARD_INFORMATION. Проводя эксперименты над файлами, авторы получают схожие результаты, что и Б. Кэрриэ в [40].

В работе [42] Cho G.-S. исследовал 8 ВО из атрибутов $STANDARD_INFORMATION и $FILE_NAME. На основе предложенной модели процесса изменения значений ВО автор предлагает рассмотреть типовые шаблоны комбинаций ВО, которые описывают совершенные файловые операции. В выводах автор указывает на одну из особенностей рассматриваемого массива данных – одна комбинация ВО может соответствовать нескольким типам файловых операций.

Матвеева В.С. в своей работе [43] уделяет внимание одному из недостатков ВО, а именно – простоте «подделки» их значений. Автор считает, что по ВО из атрибута $FILE_NAME можно определять факт искажения значений, и демонстрирует это на конкретном примере.

В работах [44, 45] авторы описывают примеры целенаправленного внесения изменений в значения ВО. В своих выводах они делают акцент на том, что специалист, проводящий исследование машинного носителя информации (МНИ), должен убедиться в корректности отметок интересующих его файлов ввиду возможности их умышленного искажения посредством т.н. «системных вызовов»

[46], например с помощью ПО TimeStomper [47] или SetMACE [48]. Minnaard W. в работе [44] выдвигает предположение, что для определения искажений в ВО могут быть использованы алгоритмы кластеризации данных, но не приводит примеров их практического применения.

Подводя итог описанию ВО, выделим достоинства и недостатки этого массива данных при использовании в рамках расследования инцидента ИБ. Среди достоинств следует выделить следующее:

1. Комбинации ВО достаточно для определения файловой операции.

2. Файловая операция может быть определена для каждого файла.

3. Информация о значениях ВО в служебной информации файла хранится длительно, до момента ее перезаписи (совершения новой файловой операции).

К недостаткам следует отнести следующее:

1. ВО подвержены искажениям как через системные вызовы, так и путем прямого редактирования области данных на МНИ. Использование ПО TimeStomper не требует прав администратора.

2. В момент совершения новой файловой операции информация о предыдущих может быть перезаписана.

3. Одна комбинация ВО может соответствовать нескольким типам файловых операций.

4. ВО не содержат признаков, характеризующих файл. Для сопоставления файла и файловой операции требуется осуществить дополнительные действия, направленные на определение указанных признаков.

Исходя из указанных достоинств и недостатков можно сделать вывод о том, что использование ВО для идентификации воздействий затруднено необходимостью верификации[3], а также определения признаков, характеризующих файл. Перейдем к рассмотрению журналов событий.

Журнал событий [49] представляет собой файл с расширением .evt (Windows XP и Server 2003) или .evtx (Windows Vista и новее), расположенный в

[3] Под верификацией массива данных будем понимать процесс выявления искажений – комбинаций параметров, имеющих отношение к файлу, возникновение которых невозможно в процессе штатного заполнения массива данными.

каталоге %WINDIR%\system32\config (Windows XP и Server 2003) или %WINDIR%\System32\winevt\Logs (Windows Vista и новее), где %WINDIR% – каталог с системными файлами Windows (по-умолчанию это C:\Windows). В журнал записывается информация о программных и аппаратных событиях различного вида. По-умолчанию в ОС Windows присутствуют несколько журналов:

- приложение (Application);
- безопасность (Security);
- установка (Setup);
- система (System);
- перенаправленные события (Forwarded events).

Информация в указанные журналы добавляется в виде т.н. записей, процедура формирования и добавления которых регламентирована [50]. У сторонних приложений, не входящих в состав ОС, существует возможность создать собственный файл журнала и сохранять записи в него, предварительно зарегистрировавшись в реестре как провайдер событий [50].

Предлагаемые журналами событий возможности позволяют использовать их в качестве основного массива данных при расследовании инцидентов ИБ. Так, зарегистрировав программу в реестре как провайдер событий, можно контролировать связанные с ней события, в т.ч. сбои в работе. Журналы событий позволяют определять факты авторизации пользователей в ОС и попытки повышения привилегий. Для того, чтобы осуществлять идентификацию воздействий на файлы на основе данных из журналов событий, необходимо настроить политику аудита на доступ к объектам, указав те файлы, воздействия на которые необходимо контролировать.

Для того чтобы обрабатывать информацию, содержащуюся в журналах событий, в записях присутствуют т.н. идентификаторы событий [51]. Рассмотрим некоторые из них, связанные с воздействиями на файлы:

- 4656 – дескриптор объекта запрошен;
- 4658 – дескриптор объекта закрыт;

- 4663 – выполнена попытка получения доступа к объекту.

Указанные идентификаторы событий позволяют фильтровать записи в журнале. Для классификации файловой операции и, соответственно, идентификации воздействия на файл используется поле записи «Операции доступа» журнала событий. Список файловых операций, определяемых с помощью журнала событий: создание, удаление, переименование, изменение содержимого, изменение служебной информации.

Xiaoyu H. и Shunxiang W. в своей работе [52] описали формат записей и предложили средство обработки записей журнала событий, которое позволяет автоматизировать процесс выгрузки необходимых записей с целью последующей их обработки и анализа в рамках расследования инцидента ИБ.

Процесс автоматизации сбора записей журналов событий рассмотрел Murphey R. в работе [53]. В выводах автор говорит о том, что после формирования из всех файлов журналов единого массива данных необходима последующая группировка записей и их классификация.

Вопросы автоматизации анализа рассмотрели Dwyer J. и Truta T.M. в [54]. Авторы оценили интенсивность появления записей с определенными идентификаторами событий, сравнили с определенными ими пороговыми значениями и сделали вывод о наличии/отсутствии аномалий в работе ОС. В тексте статьи подробно не рассматривается вопрос формирования пороговых значений для событий, что вызывает необходимость их проверки под конкретную КС.

Несмотря на указанные достоинства, использование журналов событий при расследовании инцидентов ИБ не является гарантией получения достоверного результата о выявлении этапов инцидента, т.к. у этого массива данных имеется существенный недостаток – возможность его полной/частичной очистки. Полная очистка осуществляется из штатного ПО «Просмотр событий» с обязательным оставлением записи об очищении. Частичная очистка, осуществляемая с целью противодействия специалисту, проводящему расследование инцидента ИБ, возможна с использованием ПО Winzapper [55]. Yongjian L., Peng W., Ming X.,

Ning Z. в своей работе [56] разработали алгоритмы восстановления поврежденных файлов с журналами событий, но в своих выводах делают акцент на том, что предлагаемые алгоритмы не всегда дают полноценный результат. С учетом того, что метод работы ПО Winzapper не известен, предлагаемые в [56] алгоритмы могут не восстановить утраченные записи.

Подводя итог описанию журналов событий, выделим достоинства и недостатки этого массива данных при использовании в рамках расследования инцидента ИБ. Среди достоинств следует выделить следующее:

1. Различные типы журналов событий позволяют фиксировать множество событий, происходящих в системе, связанных с процессами, пользователями, файлами и являются важным массивом данных при расследовании инцидента ИБ.

2. Имеются как встроенные в ОС, так и сторонние средства для обработки и анализа информации в журналах событий.

К недостаткам необходимо отнести следующее:

1. Журналы событий могут быть полностью или частично очищены, в т.ч. штатными средствами ОС.

2. Для получения возможности идентификации воздействий на файлы требуется настройка параметров аудита и указания файлов, воздействия на которые должны фиксироваться.

3. Записи журналов событий, имеющие отношение к воздействиям на файлы содержат только полный путь до файла и совершенную файловую операцию, но не содержат признаков, характеризующих. Для сопоставления файла и файловой операции требуется выполнить дополнительные действия, направленные на определение указанных признаков.

Исходя из указанных достоинств и недостатков, можно сделать вывод о том, что использование журналов событий для идентификации воздействий на файлы затруднено необходимостью верификации, а также определения признаков, характеризующих файл. Перейдем к рассмотрению ярлыков последних открытых файлов и списков переходов.

При первом открытии файла на чтение в каталоге «Recent», расположенном по пути AppData\Roaming\Microsoft\Windows\Recent относительно домашнего каталога каждого пользователя, создается ярлык. ОС использует ярлыки из этого каталога, чтобы отображать каждому пользователю список открывавшихся им файлов. Каждый ярлык представляет собой двоичный файл [57], из содержимого которого можно извлечь имя открывавшегося файла $N_f$, полный путь до него, время первого открытия файла и др. Совокупность всех ярлыков из каталога «Recent» можно рассматривать как массив данных, содержащих информацию о воздействиях на файлы.

H. Parsonage в своей работе [58] подробно изучил, при каких условиях изменяются данные ярлыков, и рассмотрел возможности их использования при исследовании МНИ, которое может проводиться, в том числе, в рамках расследования инцидента ИБ. Основной акцент H. Parsonage делает на ВО, которые связаны как с самим ярлыком, так и с файлом, на который ярлык ссылается. Автор отмечает три ВО, связанные с целевым файлом, которые помещаются в содержимое ярлыка: создания файла, модификации содержимого файла $C_f$, последнего доступа к файлу. На основании комбинации значений этих ВО возможно определение ФО, совершенной по отношению к целевому файлу. В своих выводах автор приводит примеры получаемой из ярлыков информации:

- MAC-адрес компьютера, где был установлен МНИ;
- факт изменения системного времени;
- перемещения файлов между томами (разделами) МНИ;
- размер целевого файла, на который ссылается ярлык;
- файловую операцию «Переименование», если она была осуществлена по отношению к целевому файлу;
- порядок открытия файлов после запуска КС.

Несмотря на то, что ярлыки в каталоге «Recent» без использования дополнительных массивов данных позволяют получить ответы на некоторые вопросы (изменялось ли системное время КС, перемещались ли файлы между

томами МНИ, какой MAC-адрес у КС, которой принадлежит исследуемый МНИ), возникающие в рамках расследования инцидента ИБ, их использование в части идентификации воздействий на файлы затруднительно. Во-первых, ярлыки из каталога «Recent» легко удалить штатными средствами операционной системы. Во-вторых, содержащиеся в ярлыках ВО не позволяют определить все множество совершенных файловых операций. В-третьих, как говорит сам H. Parsonage, существует возможность отключения функции автоматического создания ярлыков в каталоге «Recent» через параметр EditFlags ветки реестра ОС Windows HKEY_CLASSES_ROOT\ <ProgID>, где <ProgID> – тип расширения файлов. В-четвертых при последовательном открытии двух файлов с одинаковыми именами ярлык перезаписывается.

Еще один связанный с ярлыками последних открытых файлов массив данных называется JumpList (список переходов). Списки переходов представляют собой двоичные файлы, расположенные в каталогах AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations и AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations относительно домашнего каталога каждого пользователя. Согласно информации, опубликованной H. Carvey в трех частях статьи «Jump List Analysis» [59-61], список переходов представляет собой контейнер, который состоит из нескольких частей: упорядоченного списка элементов и областей данных из ярлыков последних открытых файлов.

Имя файла со списком переходов указывает на процесс, который создает ярлыки открывавшихся файлов. Механизм именования списков переходов на данный момент в открытых источниках не описан. Неполный список процессов указан в [62], рассмотрим лишь некоторые примеры:

- 9b9cdc69c1c24e2b.automaticDestinations-ms — Блокнот (64-разрядная версия);
- 290532160612e071.automaticDestinations-ms — WinRar (64-разрядная версия);
- 1b4dd67f29cb7196.automaticDestinations-ms — Проводник;

- a18df73203b0340e.automaticDestinations-ms — Microsoft Word 2016 (64-разрядная версия);
- b8ab77100df80ab2.automaticDestinations-ms — Microsoft Excel 2016 (64-разрядная версия);

E. Zimmerman в своей статье [63] опубликовал результаты по созданию автоматизированного средства обработки списков переходов с подробным описанием их структуры. Дополнительных результатов по анализу списков переходов автор не предоставил, что может быть связано с наличием исследования [58], проведенного H. Parsonage, где он уже подробно изучил содержимое ярлыка. Списки переходов не предоставляют каких-либо преимуществ перед ярлыками последних открытых файлов в процессе расследования инцидента ИБ, а являются лишь дополнительным способом их группировки и представления в ОС.

Подводя итог описанию ярлыков последних открытых файлов и списков переходов, выделим достоинства и недостатки этого массива данных при использовании в рамках расследования инцидента ИБ. Среди достоинств следует выделить следующее:

1. Различная информация, получаемая из содержимого ярлыков, позволяет ответить на вопросы, возникающие в процессе расследования инцидента ИБ.

2. Определение совершенной файловой операции и, следовательно, воздействия на файл основывается на методах и алгоритмах, применяемых к комбинации ВО.

К недостаткам необходимо отнести следующее:

1. Ярлыки и списки переходов могут быть полностью или частично очищены, в т.ч. штатными средствами ОС;

2. Хранимая в содержимом ярлыка комбинация ВО не позволяет определить все совершаемые файловые операции;

3. Принудительное создание ярлыков и списков переходов можно отключить.

Исходя из указанных достоинств и недостатков, можно сделать вывод о том, что использование ярлыков последних открытых файлов и списков переходов для идентификации воздействий на файлы затруднено необходимостью определения признаков, характеризующих файл, для получения возможности классифицировать все файловые операции. Перейдем к рассмотрению файлов службы упреждающего чтения Superfetch.

Служба Superfetch, запущенная в ОС Windows по-умолчанию (параметры EnablePrefetcher и EnableSuperFetch ветки реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters), предназначена для ускорения запуска программ [64]. Эффект ускорения достигается за счет того, что при первом запуске программ анализируются необходимые им дополнительные загружаемые ресурсы (библиотеки, текстовые и двоичные файлы произвольного содержимого), из которых формируются списки, помещаемые в файлы с расширением имени .pf. Указанные файлы расположены в каталоге %WINDIR%\Prefetch, где %WINDIR% – каталог с системными файлами Windows (по-умолчанию это C:\Windows). Совокупность pf файлов можно рассматривать как массив данных, содержащих информацию о воздействиях на файлы. При последующих запусках программ считывается содержимое соответствующего .pf файла, определяется список дополнительных загружаемых ресурсов и происходит упреждающее их чтение, чтобы к моменту запроса ресурсов программой они уже находились в оперативной памяти компьютера. С точки зрения специалиста, проводящего расследование инцидента ИБ, созданные pf файлы интересны тем, что указывают на факт и время запуска программы, которая потенциально может являться экземпляром ВПО. Дополнительно из pf файла специалист может получить имя запускаемого файла, полный путь до него, время изменения и создания pf файла и список необходимых ПО файлов.

Alsulami B., Srinivasan A., Dong H., Mancoridis S. в своей работе [65] описывают созданный ими классификатор ВПО на основе данных .pf файлов. Разработанное авторами ПО выступает в роли средства помощи принятия

решений. В своих выводах авторы указывают, что предложенный метод позволяет верно определять ВПО с вероятностью не менее 97%.

Исходя из назначения рассматриваемого массива данных наблюдается выраженная направленность на работу с ПО. Получаемая из массива данных информация позволяет классифицировать часть файловых операций, совершенных по отношению к целевому исполняемому файлу: создание, переименование, изменение содержимого, изменение служебной информации.

Подводя итог описанию файлов службы упреждающего чтения Superfetch, выделим достоинства и недостатки этого массива данных при использовании в рамках расследования инцидента ИБ. Среди достоинств следует выделить следующее:

1. pf файл позволяет определить не только целевой исполняемый файл, но и набор остальных файлов, необходимых для корректной работы ПО.

2. Определение совершенной файловой операции и, следовательно, воздействия на файл основывается на методах и алгоритмах, применяемых к комбинации ВО.

К недостаткам необходимо отнести следующее:

1. pf файлы могут быть полностью или частично очищены, в т.ч. штатными средствами ОС.

2. pf файл не позволяет классифицировать все совершаемые файловые операции.

3. pf файл создается только для исполняемых файлов, и не пригоден для идентификации воздействий на остальные типы файлов.

4. Принудительное создание pf файлов можно отключить в реестре ОС.

Исходя из указанных достоинств и недостатков, можно сделать вывод о том, что использование файлов службы упреждающего чтения Superfetch для идентификации воздействий на файлы затруднено вследствие того, что pf файлы относятся только к ПО. Перейдем к рассмотрению записей реестра.

Рассматривая реестр ОС Windows как массив данных, содержащий информацию о воздействиях на файлы, при расследовании инцидентов ИБ следует проанализировать несколько веток:

- Uninstall;
- MUICache;
- UserAssist.

Параметры, расположенные в ветке Uninstall [66] (полный путь: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall) имеют отношение к установленному ПО – в каждой ветке внутри Uninstall расположены параметры установки. Интерес для специалиста, проводящего расследование инцидента ИБ, представляют: DisplayName – название программы, InstallLocation – каталог с установленной программой и InstallDate – дата установки программы.

В ветке MUICache [67] (полный путь: HKEY_CURRENT_USER\Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache) хранится список соответствия между запускаемой программой и отображаемым в интерфейсе названием. У каждого пользователя список формируется отдельно в соответствии с перечнем ПО, которое он запускал.

Параметры ветки UserAssist [68] (полный путь: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count) указывают на время запуска ПО. В ветке хранятся полные пути до файлов ПО, закодированные с помощью алгоритма ROT13 [69]. По аналогии с веткой MUICache, у каждого пользователя список параметров формируется отдельно в соответствии с перечнем ПО, которое он запускал.

Достоинством указанных веток реестра следует считать возможность получения списка запускавшихся конкретным пользователем программ, а также время их запуска. Такая информация требуется в случае расследования инцидента

ИБ, когда у специалиста существует подозрение на работу ВПО в исследуемой ОС.

Совокупность параметров, расположенных в указанных ветках реестра, можно рассматривать (с ограничениями) как массив данных, содержащих информацию о воздействиях на исполняемые файлы. Получаемая из массива данных информация позволяет классифицировать лишь малую часть файловых операций, совершенных по отношению к целевому исполняемому файлу: создание, переименование. Указанный недостаток, помимо наличия возможности тривиального удаления параметров веток реестра, затрудняет использование рассматриваемого массива данных при идентификации воздействий на все типы файлов. Перейдем к рассмотрению журналов файловой системы NTFS.

Б. Кэрриэ в своей работе [40] рассмотрел два журнала ФС NTFS — журнал транзакций $LogFile и журнал изменений тома $UsnJrnl. Оба журнала могут быть использованы в качестве массива данных, содержащих информацию о воздействиях на файлы.

$LogFile представляет собой двоичный файл, формат которого описали G.-S. Cho и M. Rogers в работе [70]. Он предназначен для восстановления целостности структуры тома после сбоев КС. Журнал создается по-умолчанию на каждом томе с ФС NTFS и не существует возможности отключить систему ведения журнала. Содержимое $LogFile разделено на т.н. «страницы», размер которых зависит от размера тома: при размере тома до 2 Гб включительно размер страницы составляет 2 Кб, начиная с 2 Гб размер страницы становится равным 4 Кб. Каждая страница содержит заголовок и данные, которые представлены в виде записей. Существуют три вида записей: «начало транзакции» (еще ее называют «контрольная точка»), «обновление данных» и «подтверждение транзакции». G.-S. Cho и M. Rogers рассматривают механизм заполнения журнала записями: действия, которые осуществляются с томом (файлами, служебными структурами тома) записываются в $LogFile в виде т.н. «транзакций» [71], т.е. единых законченных операций. Каждое действие классифицируется числовым кодом, является элементарным и направлено на изменение некоторой области данных

тома. Цепочка из действий формирует транзакцию, которая записывается в журнал. В случае сбоя КС драйвер ФС считывает содержимое журнала и если операция не завершена, то отменяет все произведенные с томом изменения по цепочке действий.

Специалист, проводящий расследование инцидента ИБ, может извлечь из записей журнала, в частности, совершенные файловые операции: создание, удаление, переименование, изменение содержимого, изменение служебной информации и перемещение в пределах тома.

Информация, получаемая из записей журнала транзакций $LogFile, находит свое применение при решении различных вопросов, возникающих в процессе расследования инцидента ИБ. Так, G.-S. Cho в своей работе [72] предлагает алгоритмы проверки наличия искажений ВО на основе извлекаемых из записей 1-2 транзакций атрибутов $STANDARD_INFORMATION и $FILE_NAME. Автор упоминает, что в результате совершения файловой операции в атрибутах устанавливаются определенные комбинации значений ВО, что также отражено в транзакциях. Последующие искажения значений легко устранить, проанализировав записи $LogFile. В своих выводах G.-S. Cho говорит, что при использовании транзакций следует учитывать особенность $LogFile – в силу того, что размер журнала транзакций ограничен 64 Мб, в результате активности пользовательских и системных процессов происходит непрерывное его заполнение путем циклической перезаписи. В результате такой перезаписи часть данных журнала безвозвратно теряется, и содержимое транзакции оказывается неполным.

Palmbach D., Breitinger F. в работе [73] приходят к аналогичным выводам в отношении определения искажений ВО, что и G.-S. Cho в [72], и говорят, что временной интервал хранения информации в журнале зависит от интенсивности действий с томом и составляет 2-3 часа для тома с установленной ОС. Указанные сроки накладывают ограничения на применение $LogFile при несвоевременном начале расследования инцидента ИБ.

Несмотря на то, что транзакция описывает файловую операцию, в записях $LogFile не хранятся признаки, характеризующие файл. Тем не менее, эти признаки можно получить из таблицы MFT [40] по номеру LSN (LogFile Sequence Number), расположенному в записи журнала транзакции.

$UsnJrnl является двоичным файлом, содержимое которого разделено на записи. Его назначение заключается в хранении действий, совершенных по отношению к файлам. Начиная с Windows 7 журнал по умолчанию расположен на томе с установленной ОС. На несистемном томе активацию заполнения журнала можно произвести с помощью ПО fsutil [74], которое является штатным. fsutil позволяет также отключить ведение журнала $UsnJrnl, при этом не очищает сохранившиеся в нем записи, как это происходит с журналами событий ОС Windows. Авторы [73] обозначают проблему надежности данных в журнале изменений тома $UsnJrnl, тем не менее, т.к. $UsnJrnl – системный файл, то влияние на его содержимое может быть осуществлено только от имени пользователя, обладающего правами администратора и использующего стороннее программное обеспечение (fsutil не позволяет вносить произвольные изменения в журнал). По сравнению с ВО, pf-файлами и ярлыками последних открытых файлов, журнал изменений тома $UsnJrnl обеспечивает лучшую защиту от искажений содержащейся в нем информации.

По сравнению с журналом транзакций $LogFile, журнал изменений тома содержит более детализированную информацию о действиях над файлами, но регистрируются только факт работы с файлом и его причина. Собственно модифицируемые данные в $UsnJrnl не заносятся. Таким образом, $UsnJrnl не позволяет восстановить изменения ФС в случае сбоя КС.

Следует обратить внимание на то, что в журнале изменений тома фиксируются не файловые операции, а действия, совершенные по отношению к файлам. Полный перечень действий указан на сайте Microsoft [75], рассмотрим несколько примеров действий: закрытие дескриптора [76] файла, изменение состояния шифрования файла [77], создание файла, усечение данных альтернативного потока [40] и пр.

Junghoon O. в своей работе [78] приводит описание структуры файла $UsnJrnl, рассматривает примеры применения информации из журнала при исследовании МНИ и предлагает ПО для автоматизации процесса обработки записей [79]. Автор говорит, что ввиду циклической перезаписи содержимого $UsnJrnl, информация в нем хранится до 4-5 дней в отличие от журнала транзакций $LogFile, что позволяет использовать информацию из $UsnJrnl даже при несвоевременном реагировании на инцидент ИБ.

В работе Harrell C. [80] рассматривает пример исследования МНИ с целью нахождения ВПО. Для определения файлов, на которые были совершены воздействия, а также их источник, автор использовал записи $UsnJrnl. В своих выводах автор подчеркнул, что использование взаимосвязей между $LogFile, $UsnJrnl и MFT позволяет выявить все последствия инцидента ИБ, связанные с файлами.

Uijtewaal F., Prooijen J. в работе [81] предлагают модель системы, связывающей записи $LogFile, $UsnJrnl и MFT. Полученные результаты позволяют проводить взаимную верификацию указанных массивов данных и определять (при необходимости) содержимое $C_f$ и иную служебную информацию $X_f$ файлов, по отношению к которым осуществлялись воздействия — признаки, характеризующие файл, которые не фиксировались в записях $UsnJrnl.

Подводя итог описанию журналов файловой системы NTFS, выделим достоинства и недостатки этих массивов данных при использовании в рамках расследования инцидента ИБ. Среди достоинств следует выделить следующее:

1. Журналы ФС NTFS хранят информацию, позволяющую определить все файловые операции. $LogFile заполняется на каждом томе NTFS, но не содержит признаков, характеризующих файл.

2. $UsnJrnl обеспечивает долговременное хранение данных по сравнению с $LogFile, особенно при использовании на несистемном томе.

3. Данные в журналах ФС NTFS сложнее исказить по сравнению с остальными массивами данных, рассмотренными ранее.

Недостаток у журналов один — временной интервал хранения записей журналов ограничен часами/сутками в отличие от некоторых массивов данных, описанных ранее — ВО, ярлыков последних открытых файлов и списков переходов, записей реестра. Указанный недостаток влияет на корректность результатов расследования инцидента ИБ при несвоевременном на него реагировании.

Исходя из указанных достоинств и недостатков, можно сделать вывод о том, что использование журналов ФС NTFS, в частности журнала изменений тома $UsnJrnl, предпочтительно для идентификации воздействий на файлы при расследовании инцидента ИБ.

Журнал $UsnJrnl представляет собой разреженный файл, расположенный в каталоге $Extend. Он состоит из двух потоков данных: $Max и $J. Первый содержит служебные данные о журнале и не представляет интереса в рамках расследования инцидента. Второй, напротив, является для специалиста, проводящего расследование инцидента ИБ, ценным источником информации, так как состоит из набора записей об изменениях, произошедших с файлами как в отношении содержимого, так и служебной информации [75].

Записи журнала $UsnJrnl содержат в явном виде только часть параметров (компонентов вектора $V_f$), тем не менее, поле записи $R_f$ может быть косвенно использовано для определения изменений в компонентах $C_f$ и $X_f$ в рамках решения задачи верификации воздействия на файл. Формат записи представлен в таблице 1.1. Курсивом выделены поля, информация в которых позволяет идентифицировать файл и действия, осуществленные по отношению к нему.

Таблица 1.1. Формат записи журнала изменений тома $UsnJrnl

| **Смещение, байт** | **Размер, байт** | **Описание** |
|---|---|---|
| 0x00 | 4 | Размер записи |
| 0x04 | 2 | Версия записи |
| 0x06 | 2 | Версия ПО, создавшего запись |

| **Смещение, байт** | **Размер, байт** | **Описание** |
|---|---|---|
| 0x08 | 8 | *Идентификатор файловой записи* ($I_f$) |
| 0x10 | 8 | *Идентификатор родительского каталога* ($D_f$) |
| 0x18 | 8 | *Номер записи* ($U_f$) |
| 0x20 | 8 | *Временная отметка создания записи* ($T_f$) |
| 0x28 | 4 | *Идентификатор действия* ($R_f$) |
| 0x2C | 4 | Тип источника записи |
| 0x30 | 4 | Идентификатор безопасности |
| 0x34 | 4 | Атрибуты файла |
| 0x38 | 2 | Длина имени файла * |
| 0x3A | 2 | Начало имени файла в записи |
| 0x3C | * | *Имя файла* ($N_f$) |

Полный перечень идентификаторов действий представлен в описании записей [75] $UsnJrnl, рассмотрим лишь некоторые из них (таблица 1.2).

Таблица 1.2. Часть значений поля «Идентификатор действия»

| **Значение поля (в десятичной системе счисления)** | **Описание** |
|---|---|
| 1 | Содержимое файла перезаписано |
| 2 | Содержимое файла дополнено |
| 4 | Содержимое файла уменьшено |
| 256 | Создание файла |
| 512 | Удаление файла |
| 4096 | Предыдущее имя файла |
| 8192 | Новое имя файла |
| 32768 | Изменение служебной информации |
| 2147483648 | Закрытие файла |

В нескольких последовательных записях журнала наблюдаются комбинации идентификаторов, получаемые с помощью операции «побитовое ИЛИ» из значений, представленных в таблице 1.2.

## 1.4. Анализ методов классификации и кластеризации событий при расследовании инцидентов ИБ

При расследовании инцидентов ИБ возникает потребность в автоматизации процессов обработки и анализа данных из рассмотренных ранее массивов. Потребность обосновывается тем, что данные из массивов необходимо

классифицировать / спрогнозировать / сгруппировать в целях сокращения объема анализируемой информации. Даже несмотря на то, что объем данных в ярлыках последних открытых файлов и списках переходов, записях реестра и файлах службы упреждающего чтения Superfetch измеряется десятками записей, суммарный объем информации может расти нелинейно за счет потребности в одновременном исследовании нескольких КС, в т.ч. которые реализованы в виде виртуальных машин или используют несколько МНИ / томов. Следует отметить, что в отличие от ярлыков, файлов Superfetch и записей реестра, объем записей в журналах событий ОС, журналах ФС и комбинациях ВО измеряется десятками и сотнями тысяч.

Zhuge C. и Vaarandi R. в своей работе [82] указывают на необходимость применения автоматизации при анализе данных. Авторы рассматривают журналы событий Unix-подобных ОС [83] и сработок системы обнаружения атак, акцентируя внимание на объеме записей и использовании собственного алгоритма кластеризации LogCluster [84]. Makanju A., Brooks S., Zincir-Heywood A.N., Milios E.E. в работе [85] помимо упоминания необходимости применения средств автоматизации предлагают визуализировать обработанную информацию, что, по их мнению, способствует улучшению восприятия специалистом, проводящим исследование МНИ.

Oliner A., Ganapathi A., Xu W. в работе [86] наряду с несколькими проблемами, возникающими в обработке журналов событий (необходимость совместного анализа журналов, полноты ведения журналов, оценки степени субъективности результатов автоматизированного анализа), указывают на популярные при анализе данных методы главных компонент и опорных векторов (рассматриваются далее), которые могут найти применение при работе с массивами данных, в т.ч. содержащих информацию о воздействиях на файлы, в частности, журналами событий.

В работах [87, 88] авторы рассмотрели несколько методов, применяемых при анализе данных журналов событий ОС, в частности: деревья решений, метод главных компонент, нейронные сети, логистическая регрессия, метод опорных

векторов, кластерный анализ. Рассмотрим достоинства и недостатки каждого из указанных методов, а также их применение в существующих работах.

### 1.4.1. Деревья решений

Дерево решений — логический алгоритм, решающий задачи классификации и регрессии (определения зависимости между значениями известных и исследуемых параметров). Представляет собой объединение логических условий в структуру связного ациклического графа. Рассмотрим несколько работ, посвященных применению деревьев решений к анализу данных, содержащихся в различных журналах событий.

Ferreira D.R., Vasilyev E. в статье [19] показывают пример использования деревьев решений применительно к некоторым событиям процесса покупки и отгрузки товара на основании записей журнала для определения задержек. Авторы интерпретировали записи для составления предикатов и выявили «узкие места» анализируемого процесса. Вместе с тем, в выводах к работе указаны 2 недостатка:

1. Существует потребность в принятии допущений при создании предикатов по причине невозможности однозначной интерпретации совокупности записей журнала.

2. Зависимость результата от полученной структуры решающего дерева.

Sheluhin O., Osin A. в работе [25] рассмотрели совокупность журналов событий, в которых записывается информация от множества сервисов, функционирующих в рамках КС: Web-сервер, сервис мгновенной отправки сообщений, системы управления базами данных, журнал событий ОС, и др. Преобразовав записи из нескольких журналов в единый массив, авторы решают задачу поиска в нем аномалий с применением деревьев решений, метода K-ближайших соседей [89] и метода опорных векторов (будет рассмотрен далее). Худшую точность определения (Precision порядка 55%) показал метод K-ближайших соседей. Наилучшие результаты анализа получены с использованием бинарных деревьев решений – точность определения аномалий 97-99%. Столь высокие результаты являются неоспоримым достоинством метода.

Vu Q.H., Ruta D., Cen L. в статье [90] решают задачи автоматического поиска оптимальных значений параметров структуры дерева для анализа записей в журналах событий сетевых сервисов. В своих выводах авторы указывают, что *F*-мера (*F*-Score) [91] классификации событий ИБ по записям журналов достигает 95% при использовании автоматического подбора параметров деревьев. Преимущество автоматического подбора параметров заключается в уменьшении или исключении влияния субъективной оценки специалиста на этапе формирования структуры дерева.

Деревья решений могут применяться в задачах поиска аномалий на основе записей журналов событий, о чем сказано в рассмотренных ранее работах. Деревья позволяют с высокой точностью определять аномальные события КС (например, события ИБ) при обработке совокупности записей, но не обладают гибкостью в ситуациях, когда возникает новое событие, не предусмотренное структурой дерева.

В ситуациях, когда требуется решить задачу классификации, при которой одно дерево решений не дает приемлемую точность, применяются «ансамбли» деревьев, однако временные затраты на применение такого метода растут пропорционально количеству деревьев, что ограничивает применение ансамблей при отсутствии соответствующих вычислительных мощностей.

### 1.4.2. Метод главных компонент

Метод главных компонент (МГК) – один из основных методов уменьшения размерности данных [92]. В условиях множества массивов данных и наличия значительного количества записей (десятки и сотни тысяч) ценность идеи уменьшения объема анализируемой информации растет. Рассмотрим несколько работ, посвященных использованию МГК при анализе различных журналов событий.

Sathish V., Sudarsan S.D., Srini R. в статье [20] использовали МГК для предсказания отказов КС, участвующих в технологическом процессе, на основании записей журналов событий. В своих выводах к работе авторы указывают на перспективность использования МГК при прогнозировании

событий. В качестве примера приводится случай, когда удалось спрогнозировать отказ КС за 30 дней до непосредственного события. Точность прогнозирования предлагаемого авторами метода не указана в тексте работы, что не позволяет оценить эффективность на фоне других методов уменьшения размерности данных. Chuah E., Jhumka A., Alt S., Villalobos J.J., Fryman J.B., Barth W.L., Parashar M. в работе [21] также применили МГК для прогнозирования отказов высоконагруженных систем, и создали ПО «EXERMEST», которое позволяет находить взаимосвязи между событиями, полученными из записей журналов и отказами дисковой, сетевой и другими подсистемами КС.

Lalwani H., Gupta R., Srivastava S., Jayaram S. в работе [93] использовали МГК для предсказания отказов в КС на основании множества записей полученных из совокупности журналов событий. Для предварительной обработки журналов событий и формирования из записей единой матрицы, которая является входными данными МГК, использовался алгоритм SLCT [94]. Авторам удалось достичь точности (Accuracy) предсказания событий, равной 92% на тестовом наборе данных объемом 1 Тб. Вместе с тем, авторы указывают, что если объем выборки данных равен 100 Гб, то точность снижается вплоть до 81%. При 10 Гб точность предсказания равна 75%.

Рассмотренные работы имеют потенциал для исследований в области анализа событий ИБ с целью прогнозирования и предотвращения инцидентов ИБ. Вместе с тем, в открытых источниках не уделено внимание работам по классификации и группировке данных в массивах, содержащих в т.ч. информацию о воздействиях на файлы, что является актуальной задачей в рамках расследования возникших инцидентов ИБ.

### 1.4.3. Нейронные сети

Согласно определению А.И. Галушкина «нейронная сеть представляет собой высокопараллельную динамическую систему с топологией направленного графа, которая может получать выходную информацию посредством реакции ее состояния на входные воздействия» [95]. Нейронная сеть представляет собой систему соединённых и взаимодействующих между собой простых

«процессоров» — искусственных нейронов. Каждый «процессор» подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. Несмотря на кажущуюся простоту, такие процессоры вместе способны решать сложные задачи: классификации, кластеризации, помощи в принятии решений, аппроксимации, прогнозировании. Таким образом, нейронные сети являются универсальным методом анализа данных.

Нейронные сети, в отличии от традиционных алгоритмов анализа данных с фиксированной логикой работы, могут быть обучены. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения сеть способна выявлять сложные зависимости между входными и выходными данными.

С обучением нейронных сетей могут возникать сложности [96]. Во-первых, распространенной проблемой является насыщение сети — наличие больших значений сигналов внутри сети, спровоцированных неверно выбранными начальными весовыми коэффициентами нейронов. Во-вторых, входные данные следует нормировать, чтобы крутизна функции активации в окрестности точки, соответствующей входным данным, не стремилась к нулю. В-третьих, при использовании метода обучения «с учителем» не исключено внесение субъективной оценки результатов работы сети, что отрицательно скажется на значениях весовых коэффициентов. Существуют и иные проблемы, возникающие при обучении, однако их полное перечисление не является целью данной главы.

Несмотря на возникающие трудности в обучении нейронных сетей, метод активно используется при решении прикладных задач, описанных ранее. Рассмотрим применение нейронных сетей для анализа различных журналов событий.

Зуев В.Н., Ефимов А.Ю. в статье [22] рассмотрели существующие подходы к определению аномалий в событиях КС, указали их достоинства и недостатки и предложили метод для прогнозирования появления событий ИБ, связанных с вводимыми пользователем командами ОС. Авторы указывают, что их подход,

несмотря на учтенные недостатки рассмотренных работ, по-прежнему имеет ложные срабатывания, а точность прогнозирования определяется с помощью показателя абсолютной процентной погрешности [97], который обладает несколькими недостатками:

1. Не может быть использован при оценке нулевых значений рассматриваемых величин.

2. Не интерпретируется в интервале [0,1], т.к. зависит от разности значений оцениваемых величин.

3. Вносит бóльшие искажения в оценку отрицательных величин, нежели положительных.

Использование показателя абсолютной погрешности не позволяет сравнить полученные результаты применения нейронной сети в модели классификации данных с другими работами.

Lu S., Wei X., Li Y., Wang L. в работе [98] рассмотрели применение нескольких топологий нейронных сетей для анализа журнала событий в Hadoop Distributed File System [99]. Авторы преобразовали текстовые записи журнала событий в набор матриц. Полученные матрицы являются входными данными для их решающих моделей на основе нескольких топологий нейронных сетей. Объем анализируемой информации в журнале составил ~11,2 млн. записей. В своих выводах авторы указали, что достигли 96% *F*-меры определения аномалий, которые связаны с событиями ИБ с помощью сверточной нейронной сети (СНС). Полученные результаты позволяют утверждать является ли набор входных данных – записей журнала аномальным или нет.

Аналогичным образом преобразовали записи журналов событий Yuan F., Cao Y., Shang Y., Liu Y., Tan J., Fang B. в статье [27]. Авторы применили СНС и заявили о достижении их методом 95% точности (Accuracy) определения аномалий в наборе записей журнала.

Ren R., Cheng J., Yin Y., Zhan Y., Wang L., Li J., Luo C. в работе [100] также использовали СНС для классификации событий на основе записей журналов. Авторы использовали аналогичный [98] подход, но сократили размерность

матриц событий до векторов при преобразовании текста из записей журнала в числовые признаки. Полученные результаты рассмотрены в сравнении с остальными методами анализа данных и сделан вывод о том, что СНС демонстрирует наилучшую точность классификации (*F*-мера порядка 98%). Вместе с тем, при тех же входных данных, деревья решений и метод опорных векторов показали сравнимые результаты (*F*-меры 97% и 95% соответственно) при меньших временных затратах на подбор оптимальных параметров.

СНС, наряду с несколькими достоинствами [101] обладает и недостатком: подбор множества параметров сети не описан алгоритмически. Применяемые методы, например МГК или кластерный анализ, лишь отчасти помогают задать корректные параметры, поэтому авторы трудов по использованию СНС применяют в т.ч. эмпирический подбор значений исходя из собственной оценки результатов работы сети, внося «человеческий фактор» в итоговую оценку.

Yen S., Moh M., Moh T.-S. в статье [28] применили последовательное включение СНС и рекуррентной нейронной сети (РНС) [102] для анализа данных журналов событий. С помощью СНС авторы сократили объем и размерность признаков анализируемой информации, с помощью РНС классифицировали данные на предмет наличия или отсутствия аномалий в записях журнала. В выводах авторы указывают, что их подход примерно в 2 раза уменьшает временные затраты на 1 раунд обучения (с 28 секунд до 15 секунд) по сравнению с РНС типа LSTM при аналогичной точности классификации и увеличивает точность (*F*-мера) на 1,5% по сравнению с СНС при сопоставимых временных затратах на 1 раунд обучения. Недостатки подхода следуют из недостатков применяемых типов нейронных сетей: неоднозначность процесса подбора параметров СНС и склонность РНС к переобучению [103].

Рассмотренные работы позволяют сделать вывод о широкой применимости нейронных сетей при анализе массивов данных. Отличительной чертой метода является возможность обучения – установки значений весов нейронов, позволяющей настроить сеть на решение прикладных задач в условиях, когда входные данные, на первый взгляд, не имеют явных взаимосвязей. Вместе с тем,

работы, посвященные анализу записей журналов событий, имеют узкую направленность: продемонстрированные решения позволяют определить лишь факт наличия аномальных данных (в некоторых работах без конкретизации значений слова «аномальность» и того, что под ним подразумевается), но не указывают ее тип. Не вводится понятие «нормальность», которое позволило бы однозначно отделить штатные события от аномальных, но указывается высокая точность нахождения аномалий в массиве данных.

Решение вопросов идентификации воздействий на файлы с помощью нейронных сетей является перспективным направлением для исследований у специалистов по ИБ, которые по роду деятельности принимают участие в расследовании инцидентов ИБ. Несмотря на универсальность метода, при решении прикладных задач классификации, кластеризации, предсказания нейронные сети не всегда показывают наилучшие результаты по сравнению с традиционными алгоритмами, рассматриваемые в данной главе. Проблема переобучения нейронной сети не всегда позволяет оперативно изменить параметры сети при наличии изменений формата входных данных.

### 1.4.4. Логистическая регрессия

В математической статистике логистическая регрессия [104] является статистической моделью, которая использует логистическую функцию для моделирования зависимости выходной переменной от набора входных. Выходная переменная принимает значения 0 или 1.

Важность логистической регрессии обусловлена тем, что многие задачи анализа массивов данных могут быть решены с помощью бинарной классификации или сведены к ней. Например, с помощью логистической регрессии можно оценивать вероятность наступления (или не наступления) событий. Благодаря этому логистическую регрессию можно рассматривать как инструмент поддержки принятия решений. Рассмотрим несколько работ, посвященных применению логистической регрессии.

Alji M., Chougdali K. в работе [105] использовали бинарную логистическую регрессию для определения факта искажений ВО. Авторы получили 95% точность

(Precision) на «синтетических» тестовых данных в лабораторных условиях. Предлагаемый авторами подход не позволяет установить корректное значение искаженных ВО.

Berlin K., Slater D., Saxe J. в статье [29] применили логистическую регрессию для определения активности ВПО по данным из записей журнала событий ОС Windows. Авторы рассматривали журналы, содержащие от 300 до 600 тысяч записей. Для структурирования значимых признаков из записей журналов использовались матрицы. Отличительной особенностью работы является сравнение точности определения ВПО предлагаемого метода с существующими на рынке продуктами от фирм Kaspersky, Symantec и McAfee. В выводах авторы указали, что точность (Precision) определения их модели на основе логистической регрессии составила 85%. В сравнении с существующими на рынке продуктами, авторский метод обнаружил ВПО в 80% случаев, которые антивирусы посчитали не опасным ПО.

Qin T., Gao Y., Wei L., Liu Z., Wang C. в статье [106] изучили проблему определения аномальных событий по множеству разноформатных журналов. Предложенная модель анализа данных основана на логистической регрессии. Авторы указывают на то, что различие в записях множества журналов оказывает существенное влияние на точность определения аномалий и приводят свой метод предварительной обработки данных. В выводах к работе указано сравнение полученных результатов с алгоритмом *k*-средних (рассмотрен далее), деревьями решений и наивным классификатором Байеса [89]. Наилучший результат (*F*-мера 82%) показал метод логистической регрессии.

Kaiafas G., Varisteas G., Lagraa S., State R., Nguyen C.D., Ries T., Ourdane M. в статье [107] сравнили деревья решений и логистическую регрессию при поиске аномальных событий, связанных с авторизацией пользователей на основе записей журналов событий. Авторы предлагают 2 модели: модель данных для формирования определяющих событие признаков, которая будет являться входными данными для модели принятия решений. В выводах указано, что

модель принятия решений на основе логистической регрессии позволила получить точность (*F*-мера) 65% при определений аномальных событий.

В рассмотренных работах описано применение логистической регрессии при определении аномалий (аномальных событий) в массивах данных, в т.ч. связанных с идентификацией воздействий на файлы. Тем не менее, представленные решения сводятся к определению факта аномального события без указания его типа и времени возникновения. Указанная особенность затрудняет применение логистической регрессии при анализе массивов данных, содержащих информацию о воздействиях на файлы в рамках расследования инцидентов ИБ.

### 1.4.5. Метод опорных векторов

Метод опорных векторов – один из популярных методов анализа данных, который применяется для осуществления классификации и регрессии. Его суть состоит в задании параметров т.н. «гиперплоскости», разделяющей объекты выборки оптимальным способом – чем больше расстояние между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет ошибка классификатора [108, 109].

В статье [110] Jayan K., Rajan A.K. применили метод опорных векторов (SVM) для анализа журнала событий межсетевого экрана в целях уменьшения количества обрабатываемой информации путем ее классификации на «аномальную» и «нормальную». Авторы не привели критериев аномальности, но смогли уменьшить до 2.5 раз количество записей журнала (со 100 тыс. до 40 тыс. и с 700 тыс. до 325 тыс.), которые необходимо проанализировать при расследовании инцидента ИБ и внедрили свой метод в существующую систему защиты информации.

Шелухин О.И. и Рябинин В.С. в статье [26] сравнили несколько методов анализа данных в отношении классификации записей журналов событий суперкомпьютера BlueGene/L [111]. Подход авторов к подготовке входных данных схож с рассмотренными ранее работами и представляет собой преобразование текстовых строк в матрицу признаков, которая подается на вход решающей модели, использующей несколько методов анализа данных. В качестве

основы для решающей модели использовались метод опорных векторов, деревья решений и K-ближайших соседей. В выводах авторы указали, что наибольшую точность (Precision 99%) обеспечивает модель на основе SVM, деревья решений обеспечили Precision 92%. K-ближайших соседей не достиг 90% значения Precision в определении аномалий.

Akanle M., Adetiba E., Akande V., Akinrinmade A., Ajala S., Moninuola F., Badejo J., Adebiyi E. в статье [112] применили несколько базисных функций для реализации SVM в целях классификации событий по записям журналов. Подход авторов к подготовке входных данных заключался в структурировании текстовых записей журналов событий в виде матриц, аналогично ранее рассмотренным работам. В выводах к статье заявлена 100% точность (Precision) в определении аномалий при использовании линейной и радиальной базисных функций. По мнению авторского коллектива, при решении задачи классификации событий на основе записей журналов предпочтительно использовать SVM вместо нейронных сетей в силу того, что процесс обучения нейронной сети требует бóльших временных затрат, нежели подбор параметров SVM.

В рассмотренных работах авторы акцентируют внимание на вопросах классификации записей журналов событий. SVM обеспечивает высокую точность (вплоть до 100%) определения аномалий в зависимости от структуры записи журнала. Вместе с тем, метод имеет несколько недостатков [113]:

1. Неустойчивость к шуму: «выбросы» в исходных данных становятся опорными объектами и напрямую влияют на построение разделяющей гиперплоскости, а следовательно, точность классификации и регрессии.

2. Не описаны общие методы выбора базисных функций, наиболее подходящих для конкретной задачи.

3. Не позволяет осуществить группировку данных.

С учетом указанных недостатков метод опорных векторов затруднительно применять при анализе массивов данных, содержащих информацию о воздействиях на файлы в рамках расследования инцидентов ИБ.

### 1.4.6. Кластерный анализ

Кластерный анализ – это «способ группировки многомерных объектов, основанный на представлении результатов отдельных наблюдений точками подходящего геометрического пространства с последующим выделением групп как «сгустков» этих точек» [114].

Кластерный анализ находит свое применение в тех случаях, когда исходные данные представлены в виде матриц расстояний между объектами либо в виде точек в многомерном пространстве. Результатом анализа является выявление и группировка некоторых геометрически удаленных групп объектов, внутри которых эти объекты близки [115]. Рассмотрим несколько работ посвященных кластерному анализу.

Vaarandi R. в работе [30] представил разработанный алгоритм кластеризации, реализованный в виде ПО Simple Log Clustering Tool (SLCT). SLCT предназначен для кластеризации данных, полученных из текстовых массивов, например журналов Syslog [83]. Автор реализовал предварительное разбиение текстовых записей на признаки, из которых формируется информация, подаваемая на вход разработанного алгоритма кластеризации. ПО SCLT легло в основу дальнейших исследований специалистов по обработке и анализу данных из журналов событий, в частности [85, 93].

Makanju A., Zincir-Heywood A.N., Milios E.E. в работе [116] предложили свой алгоритм кластеризации данных, полученных из журналов, записи которых являются текстовыми строками — Iterative Partitioning Log Mining (IPLoM). Необходимость в создании нового алгоритма авторы обосновали недостаточной производительностью ранее представленного SLCT. В тексте работы не приводятся абсолютные значения производительности, из которых сделан вывод о ее недостаточности, в том числе в сравнении SLCT и IPLoM. В выводах указано, что значения *F*-меры у IPLoM превышает таковой у SLCT более чем в два раза (30% против 80%).

Vaarandi R. с соавторами в статьях [31, 32] предложили новый алгоритм LogCluster для кластеризации записей журналов Syslog. По словам авторов, в

новом алгоритме применена иная процедура преобразования текста в признаки. Исходя из результатов апробации, представленных в статьях, LogCluster проигрывает SLCT во временных затратах на обработку записей массива данных (например, 3050 против 1911 секунд). Представленные результаты не позволяют оценить преимущество LogCluster перед ранее разработанным алгоритмом SLCT.

Stein B., Niggemann O. в статье [33] предложили алгоритм кластеризации MajorClust, основанный на представлении данных в виде неориентированного взвешанного графа и объединения точек, представляющих данные, в кластеры на основе подсчета суммарного веса ребер смежных точек. Авторы указали три направления применения алгоритма: мониторинг компьютерных сетей на основе анализа пакетов сетевого трафика, визуализация данных, представленных сложными графами, путем разделения их на более простые, группировка данных в многомерном пространстве на основе значений метрик. Представленный алгоритм обладает несколькими достоинствами. Например, автоматически определяет необходимое количество кластеров и их размер. Развитием идей MajorClust явилась работа [34], в которой авторы устранили склонность алгоритма к объединению кластеров с соизмеримыми весами ребер, соединяющих близко расположенные вершины. Заявленная точность (Accuracy) при формировании кластеров оценена авторами в 83%.

Другую идею кластеризации на основе графов реализовали Studiawan H., Pratomo B.A., Anggoro R. в статье [35]. Авторы предложили механизм формирования кластеров на основе фильтрации найденных $k$-клик графа [117], где $k$ – количество вершин в рассматриваемой клике. В выводах указано, что предложенный алгоритм обладает недостатком, а именно зависимостью скорости работы от объема входных данных. Дополнительно следует отметить, что пример, демонстрирующий результаты работы алгоритма, не отличается от результатов фильтрации по ключевым словам.

Nagappan M., Vouk M. в статье [118] реализовали идею о том, что записи журналов событий можно кластеризовать на основе частоты появления в них слов. В своем алгоритме авторы разделили структуру записей на постоянные

типы событий (словосочетания) и переменные данные (слова), добавив при этом взаимосвязь на основе частоты появления слов. Таким образом, авторы применили шаблоны типов событий, в виде словосочетаний, ассоциированные с переменными данными.

Результаты, полученные в рассмотренных выше работах позволяют выделить несколько достоинств кластерного анализа:

1. Допускается разбиение данных на кластеры не по одному-двум параметрам, а по совокупности нескольких параметров;

2. Нет ограничений на вид анализируемых данных (числа, символы, векторы и др.);

3. Возможно циклическое использование кластерного анализа до достижения необходимого результата разбиения.

Вместе с тем, как и у других ранее представленных методов, у кластерного анализа имеются и недостатки:

1. При объединении в кластеры могут «теряться» признаки, характерные для одиночных данных.

2. Состав и количество кластеров определяется критерием разбиения (параметрами алгоритма), который подбирается эмпирически.

Второй недостаток следует напрямую из формулировки теоремы Клейнберга [119] «Для множества из двух и более элементов, не существует алгоритма кластеризации, который бы одновременно удовлетворял 3 условиям: масштабно-инвариантный, полный и согласованный», т.е. алгоритм кластеризации должен быть независимым от функции расстояния, разделять выборку на любое фиксированное количество кластеров для выбранного критерия, и не зависеть от изменения меж- и внутрикластерного расстояния для заданного критерия разбиения *одновременно*. Тем не менее, несмотря на отсутствие возможности выбора оптимального алгоритма кластеризации, неоднозначности в разбиении могут быть устранены экспертом на основании оценки им результатов.

С учетом рассмотренных работ по анализу журналов событий, указанных достоинств и недостатков метода, кластерный анализ является предпочтительным методом уменьшения объема обрабатываемой информации при расследовании инцидентов ИБ.

### 1.4.7. Выбор алгоритма для анализа записей журнала изменений тома $UsnJrnl

Для реализации процесса автоматизированного объединения нескольких записей журнала в совокупность с целью получения воздействия на файл возможно использование различных алгоритмов кластерного анализа, например, *k*-средних, *k*-средних++, *k*-медоидов, DBSCAN. Упоминавшиеся ранее алгоритмы кластеризации SLCT, LogCluster, IPLoM нацелены на работу с текстовой информацией, получаемой из записей журнала Syslog, и не подходят для решения текущей задачи. Алгоритмы на основе дендрограмм, графов и иерархическом подходе не позволили корректно разделить объекты, размещение на плоскости которых подчинено рассматриваемой в п. 2.2.1 зависимости на кластеры.

Алгоритм *k*-средних [120] является итерационным алгоритмом, принцип работы которого заключается в минимизации суммарного квадратичного отклонения точек кластеров от их центров – на каждой итерации пересчитывается «центр масс» для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь и так повторяется до тех пор, пока на какой-то итерации не происходит изменения кластеров. Вычислительная сложность алгоритма составляет $O(kid'n')$, где $k$ – количество кластеров, $i$ – количество итераций алгоритма, $d'$ – размерность вектора входных данных, $n'$ – количество точек.

Алгоритм *k*-средних++ [121] является модификацией *k*-средних. Отличие заключается в том, что на начальном этапе не все центры кластеров выбираются произвольно, а только один. Остальные центры рассчитываются исходя из значения квадрата расстояния от точки до ближайшего центра. Вычислительная сложность алгоритма, по заявлению авторов, ниже таковой у *k*-средних и составляет $O(k)$, где $k$ – количество кластеров. *k*-средних++ позволяет

минимизировать ошибку в изначальном выборе центра кластера и ускорить время выполнения, что актуально для задач с объемом входных данных от нескольких тысяч точек. Результаты проведенного исследования относительно различий в скорости выполнения *k*-средних и *k*-средних++ на данных с объемом в 10000 точек совпадают с таковыми в [121].

Алгоритм *k*-медоидов [122] похож на *k*-средних, одно из ключевых отличий заключается в порядке выбора центра кластера – центром выбирается одна из точек кластера, а не его «центр масс». В связи с особенностью выбора центра кластера, алгоритм *k*-медоидов показал отрицательные результаты в тех случаях, когда точки в пространстве распределены в форме четверти окружности. Такие ситуации возникают, когда интенсивность нескольких простых воздействий на файл невысока, что соответствует особенностям зависимости, рассматриваемой в п. 2.2.1. Вычислительная сложность алгоритма составляет $O(n^2k^2)$, где $n$ – количество точек, $k$ – количество кластеров.

DBSCAN [123], несмотря на возможность работы с кластерами произвольной формы, требует точного задания двух параметров: $\varepsilon$ и minPts, которые подбираются эмпирически для исследуемого набора данных. В случае неоптимального задания этих параметров данные либо не будут кластеризованы, либо сольются в один кластер. Одиночные точки помечаются как «шум», в то время как они могут являться отдельным кластером. Алгоритм показывает наилучшее разбиение на кластеры в случаях, когда данные распределены в виде концентрических окружностей, полуокружностей и иных произвольных формах, которые не встречаются в рамках исследуемой в п. 2.2.1 зависимости. Модификация алгоритма, например, Generalized DBSCAN позволяет не указывать $\varepsilon$ и minPts, но уступает в скорости работы по сравнению с *k*-средних++ и *k*-медоидов в рамках решаемой задачи. Вычислительная сложность алгоритма составляет $O(n\log_2 n)$ в случае отсортированных входных данных и $O(n^2)$ на необработанных входных данных, где $n$ – количество точек.

Для автоматизации процесса идентификации воздействий на файлы в рамках настоящего исследования был выбран алгоритм кластеризации

*k*-средних по соотношению скорость работы (на количестве точек, не превышающем 1000 в рамках предлагаемого метода) / сложность подготовки входных данных и оценке полученных результатов. Несмотря на то, что наилучшие результаты алгоритм достигает при кластеризации скоплений точек сферической формы, в рамках исследования получены удовлетворительные результаты без использования методов спектральной кластеризации [124], в частности *k*-средних с нелинейным ядром [125].

Сравнение результатов по скорости работы указанных алгоритмов происходило с использованием ПО MATLAB и представлено в таблице 1.3.

Таблица 1.3. Сравнение скорости работы алгоритмов кластеризации

| **Алгоритм** | **Затраченное время на выполнение кластеризации, с** | | | | |
|---|---|---|---|---|---|
| | **50 точек** | **200 точек** | **500 точек** | **1000 точек** | **10000 точек** |
| DBSCAN | 0,004 | 0,010 | 0,025 | 0,061 | 4,475 |
| k-медоидов | 0,012 | 0,031 | 0,357 | 2,766 | 7,498 |
| k-средних | 0,007 | 0,009 | 0,017 | 0,032 | 1,406 |
| k-средних++ | 0,008 | 0,015 | 0,024 | 0,050 | 0,706 |

Необходимым условием работы алгоритма *k*-средних является указание желаемого количества кластеров *k* для группировки входного множества значений. От выбора значения *k* зависит корректность получаемых результатов. Пример задания верного количества кластеров указан на рисунке 1.2, где видно, что точки в пространстве были сгруппированы и обозначены цветом оптимальным образом. При установке значения *k*, отличного от 3, для группировки тех же данных, полученный результат окажется некорректным.

Существуют несколько методов, позволяющих определить оптимальное значение *k*: метод «локтя», метод «силуэтов». Несмотря на то, что указанные методы популярны при определении *k*, они допускают неоднозначное решение задачи определения оптимального *k*. Рассмотрим недостатки указанных методов.

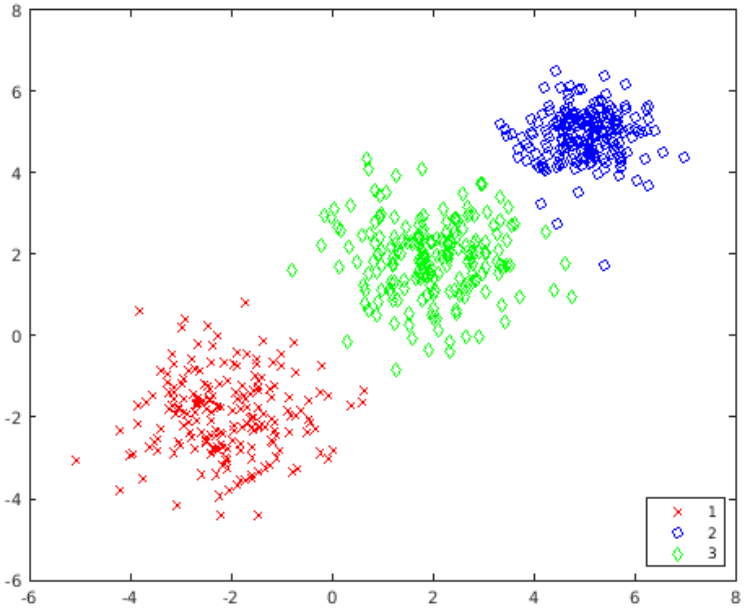


Рисунок 1.2. Группировка точек на плоскости в $k = 3$ кластера

Метод «локтя» [126] основан на расчете суммы квадратов расстояний между точками кластера и их центрами при $k = \overline{1, M}$, $M$ – количество точек в блоке. Построенный график зависимости позволяет определить оптимальное значение $k$ в «локте» – месте, где крутизна графика уменьшается на наибольшую величину по сравнению со всеми остальными значениями $k$. Метод обладает недостатком: если исходные данные распределены таким образом, что график зависимости будет монотонно убывать, то определение оптимального $k$ станет невозможным в силу одинаковых значений крутизны.

Метод «силуэтов» [127] основан на расчете коэффициента, равного $\frac{b_k - a_k}{\max(b_k, a_k)}$, где $a_k$ – среднее расстояние между всеми точками в кластере, где $b_k$ – среднее расстояние между всеми точками кластера и точками в ближайшем соседнем кластере, $k = \overline{1, M}$, $M$ – количество точек. Коэффициент силуэта принимает значение в диапазоне от -1 до 1, причем чем ближе к 1, тем лучше полученный результат кластеризации. Метод обладает недостатком: при последовательном увеличении количества кластеров требуется пересчет ранее вычисленных коэффициентов, т.к. их значения уменьшаются. При некотором значении $k$ может возникнуть ситуация когда несколько коэффициентов «силуэта» станут близки по значению, т.е. разница между стремится к 0. В таком

случае сделать вывод об оптимальном значении $k$ затруднительно в силу того, что полученное разбиение на кластеры не является корректным.

Ввиду наличия вышеуказанных недостатков у распространенных методов, в качестве основы решения задачи выбора оптимального значения $k$ использован принцип минимальной длины описания (Minimum Description Length) [128, 129]. Идея MDL заключается в следующем: «Любая закономерность в заданном наборе данных может быть использована для сжатия данных, то есть описания данных с использованием меньшего набора символов, чем нужно для описания данных буквально» [129], т.е. осуществить «сжатие» информации. В роли количественной оценки величины, которая необходима для описания набора данных, выступает длина описания $L(x)$. В соответствии с принципом MDL, нахождение минимального значения $L(x)$ позволяет определить оптимальный способ описания данных. Под набором данных понимаются точки на плоскости, связанные с записями журнала изменений тома \$UsnJrnl, а закономерностью является принадлежность точек кластерам. В соответствии с принципом MDL, ожидаемым результатом является представление точек на плоскости кластерами, количество которых меньше, чем точек.

В работе [130] Rissanen J. показал, что, основываясь на формуле информационного критерия Шварца [131], расчет длины описания исследуемых случайно распределённых величин с помощью предлагаемой модели осуществляется по формуле:

$$L(x) = -\sum_{x \in X} \log_2 p(x) + \frac{1}{2} P \log_2 |X| . \qquad (5)$$

Первым слагаемым является отрицательное значение логарифмической функции правдоподобия для всех $x \in X$, где $X$ – вектор исследуемых значений. Второе слагаемое – «штраф» за увеличение количества оцениваемых параметров $P$ модели. Для того, чтобы рассчитать оптимальное значения $k$ необходимо будет внести изменения в формулу (5), которые описаны в п. 2.2.3.

### 1.4.8. Обоснование необходимости разработки новых методов анализа данных в рамках решения задач идентификации, обнаружения и классификации воздействий на файлы

В рамках решения задач по анализу воздействий на файлы для автоматизации расследования инцидентов ИБ необходимо использовать как методы уменьшения размерности данных, так и методы классификации.

Описанные в п.1.4.7 алгоритмы кластерного анализа могут быть применены для сокращения объема анализируемой информации, но требуется разработка метода автоматического подбора параметров кластеризации, который будет основан на указанном выше принципе MDL, т.к. существующие методы автоматического подбора параметров не позволяют точно определить оптимальные значения параметров.

Сложности, возникающие при применении алгоритмов классификации, связаны с уменьшением точности получаемых результатов ввиду учета незначимых признаков в анализируемой информации, а также с ограниченным использованием методов для определения лишь наличия аномалий в массиве данных. Любое изменение в наборе признаков влечет за собой процедуру переобучения классификатора, которая может оказаться ресурсоемкой и затратной по времени.

Рассмотрим на примере журнала изменений тома $UsnJrnl проблемы, связанные с применением существующих методов классификации. На рисунке 1.3 показана часть записей журнала в виде таблицы. Столбцы таблицы соответствуют следующим полям записей журнала (таблица 1.1):

- столбец «ИФЗ» содержит идентификаторы файлов $I_f$;
- столбец «ИРК ФЗ» содержит идентификатор родительских каталогов $D_f$;
- столбец «Операция» содержит идентификаторы действий $R_f$, совершенных по отношению к файлам;
- столбец «Имя» содержит имена файлов $N_f$;
- столбец «USN» содержит номера записей $U_f$;
- столбец «Временная отметка» содержит время создания записей $T_f$.

| ИФЗ | ИРК ФЗ | Операция | Имя | USN | Врем. отметка |
|---|---|---|---|---|---|
| 52067 | 3210 | 256 | Microsoft-Windows-Backup.evtx | 568192 | 131421909577972226 |
| 52067 | 3210 | 258 | Microsoft-Windows-Backup.evtx | 568312 | 131421909578128226 |
| 52067 | 3210 | 259 | Microsoft-Windows-Backup.evtx | 568432 | 131421909578128226 |
| 21517 | 21635 | 4 | c.wnry | 569552 | 131421909608548279 |
| 21517 | 21635 | 6 | c.wnry | 569624 | 131421909608548279 |
| 21517 | 21635 | 2147483654 | c.wnry | 569696 | 131421909608548279 |
| 52036 | 465 | 4 | @WanaDecryptor@.bmp | 570080 | 131421909611980285 |
| 52036 | 465 | 2147483652 | @WanaDecryptor@.bmp | 570184 | 131421909611980285 |
| 52036 | 465 | 2 | @WanaDecryptor@.bmp | 570288 | 131421909611980285 |
| 52036 | 465 | 3 | @WanaDecryptor@.bmp | 570392 | 131421909611980285 |
| 52036 | 465 | 32771 | @WanaDecryptor@.bmp | 570496 | 131421909612136286 |
| 52036 | 465 | 2147516419 | @WanaDecryptor@.bmp | 570600 | 131421909612136286 |
| 52052 | 51800 | 4 | @WANADECRYPTOR@.EXE-72508EF8.pf | 572392 | 131421909710728459 |
| 52052 | 51800 | 6 | @WANADECRYPTOR@.EXE-72508EF8.pf | 572520 | 131421909710728459 |
| 52052 | 51800 | 2147483654 | @WANADECRYPTOR@.EXE-72508EF8.pf | 572648 | 131421909710728459 |
| 20080 | 13650 | 1 | thumbcache_96.db | 572776 | 131421909710884459 |
| 52071 | 52049 | 256 | state.tmp | 572872 | 131421909790756599 |
| 52071 | 52049 | 258 | state.tmp | 572952 | 131421909790756599 |
| 52071 | 52049 | 2147483906 | state.tmp | 573032 | 131421909790756599 |
| 52051 | 52049 | 2147484160 | state | 573112 | 131421909790756599 |
| 52071 | 52049 | 4096 | state.tmp | 573184 | 131421909790756599 |
| 52071 | 52049 | 8192 | state | 573264 | 131421909790756599 |

Рисунок 1.3. Часть записей загруженного журнала изменений тома $UsnJrnl

Специалист для обучения классификатора выбирает значимые на его взгляд признаки (в рассматриваемом случае – столбцы таблицы) и «размечает» данные – классифицирует записи. В рассматриваемом примере можно классифицировать записи, связанные с: заполнением журнала событий (выделены синим цветом), запуском и работой ВПО WannaCry (выделены красным цветом), созданием pf файлов службы упреждающего чтения Superfetch после запуска ВПО WannaCry (выделены фиолетовым цветом), изменение и переименование временного файла state.tmp (выделены зеленым цветом). Если учитывать все признаки (столбцы) при

обучении классификатора, то при работе алгоритма на целевой системе, где произошел инцидент ИБ, возникнет проблема, связанная с тем, что файлы с такими же именами могут иметь другие идентификаторы файлов $I_f$ и идентификаторы каталогов $D_f$, т.к. $I_f$ и $D_f$ задаются драйвером ФС в процессе функционирования КС. Аналогично номера записей $U_f$ и временные отметки появления записей $T_f$ могут отличаться. В результате точность классификации будет ниже ожидаемой. Вместе с тем, исключив из рассмотрения $I_f$, $D_f$, специалист ограничит возможности алгоритма по классификации воздействий на определенные файлы целевой КС, представляющие интерес с позиции ИБ: файлы конфигурации ОС, исполняемые файлы сервисов КС и т.д.

Следует учитывать, что в процессе расследования инцидента ИБ возникают ситуации, когда нужно классифицировать воздействия на определенные типы файлов, которые характеризуются в бóльшей степени расширением, нежели именем. Тем не менее, предусмотреть все возможные комбинации имен и расширений файлов не представляется возможным, в связи с чем вновь возникают ограничения при использовании существующих методов и алгоритмов классификации.

Таким образом, в процессе классификации воздействий на файлы следует динамически определять значимые признаки, по которым анализируется информация, в целях повышения точности получаемых результатов. Решением этой задачи может служить совокупность данных, которая является эталоном при сравнении значений признаков. В рамках работы такие данные будем называть шаблоном воздействия на файл, который представляет собой универсальную структуру, учитывающую только значимые признаки воздействия в целях повышения точности процедуры классификации.

В связи с введением новой структуры данных возникает потребность в разработке метода, который позволяет проводить классификацию воздействий на файлы с учетом имеющихся шаблонов и обеспечивать экспресс-анализ событий ИБ.

## 1.5. Методы моделирования на основе сетей Петри

В качестве инструмента формализации алгоритмов и процессов используются несколько типовых схем [132]:

- непрерывно-детерминированные модели (*D*-схемы) – используются для изучения динамики моделируемой системы во времени;
- дискретно-детерминированные модели (*F*-схемы) – используются для описания дискретных состояний и дискретного характера работы моделируемой системы во времени. В качестве основы применяются конечные автоматы. Тем не менее, этот вид схем непригоден для описания процессов принятия решений, процессов в динамических системах с наличием переходных процессов;
- дискретно-стохастические модели (*P*-схемы) – используются подобно *F*-схемам, но с применением вероятностных автоматов;
- непрерывно-стохастические модели (*Q*-схемы) – используются для описания систем массового обслуживания;
- сетевые модели (*N*-схемы) – используются для моделирования процессов, связанных с «формализованным описанием и анализом причинно-следственных связей в сложных системах» [132]. Примером математического аппарата, применяемого в качестве основы для *N*-схем, являются сети Петри [133];
- комбинированные модели (*A*-схемы) – универсальные виды схем, которые используются для моделирования систем, не описываемых указанными выше схемами.

Для формализации процесса определения признаков, характеризующих файл, и принятия решения об осуществленном воздействии необходимо учитывать дискретный характер их изменения.

Математический аппарат сетей Петри хорошо зарекомендовал себя [134] при моделировании процессов, происходящих в КС, т.к. позволяет описать сложную природу КС, в которой выполняются различные по типу и назначению процессы: последовательные и параллельные, синхронные и асинхронные.

Сеть Петри, которая позволяет осуществить принятие решений, должна обладать т.н. маркировкой – способом определения порядка задействования переходов. Такая сеть задается пятью параметрами — *P, T, E, Q, μ* [133], где:

- *P* – множество позиций;
- *T* – множество переходов;
- *E* – входная функция;
- *Q* – выходная функция;
- *μ* – маркировка.

Для моделирования процесса идентификации воздействий на файлы необходимо дополнить стандартную модель на основе сети Петри как показано в п. 2.1.3.

## 1.6. Известные программные средства, применяемые при расследовании инцидентов ИБ

В настоящее время представлены несколько видов ПО, которые могут быть использованы для расследования инцидентов ИБ. Некоторые существующие виды ПО, рассмотренные в рамках данной работы, представлены в таблице 1.4.

Таблица 1.4. Пример существующего ПО, используемого при расследовании инцидентов ИБ

| Наименование | Версия | Разработчик | Лицензия |
|---|---|---|---|
| Autopsy | 4.16.0 | Brian Carrier, Basis Technology | Свободно распространяемое ПО |
| Belkasoft Evidence Center | 9.5 | ООО «Белкасофт» | Коммерческое ПО (демонстрационная версия) |
| LastActivityView | 1.30 | Nir Sofer | Свободно распространяемое ПО |

ПО Autopsy предназначено для проведения криминалистических исследований МНИ. Среди основных возможностей выделяют следующие:

- поиск файлов по сигнатурам (последовательности байт, характеризующей тип файла);
- контекстный поиск файлов;
- построение временной линии событий из найденных файлов и записей журналов;

- загрузка записей журналов событий;
- загрузка данных из pf файлов службы Superfetch;
- загрузка списка последних открытых файлов из каталога Recent;
- обработка ВО файлов;
- поиск и восстановление удаленных файлов;
- поиск сообщений электронной почты;
- поиск данных в служебной информации изображений;
- обработка и анализ данных ОС Android;
- чтение сообщений локальных БД мессенджеров WhatsApp, Viber, Hangouts и др. на образе МНИ с ОС Android;
- и др.

ПО Belkasoft Evidence Center также используется для проведения криминалистических исследований МНИ. В функциональные возможности ПО входит список, аналогичный у Autopsy, а также добавлены дополнительные возможности по анализу текстовой информации на изображениях, построению графа событий ОС и др.

ПО LastActivityView из набора программ NirSoft предназначено для сбора и обработки информации из нескольких массивов данных (записи журналов событий, записи реестра Windows, pf файлы, каталог последних открытых файлов Recent) с целью отображения действий пользователя. ПО позволяет отобразить конечный список типов действий, в котором, в частности, присутствуют:

- запуск исполняемого exe файла;
- открытие файла или каталога;
- открытие каталога в «Проводнике»;
- установка ПО;
- запуск/выключение ОС;
- авторизация пользователя.

Указанные типы действий относятся к информации, с использованием которой можно в ручном режиме идентифицировать и обнаружить некоторые простые воздействия на файлы при расследовании инцидентов ИБ.

Belkasoft Evidence Center и Autopsy являются универсальными средствами для решения широкого спектра задач по поиску и анализу информации, возникающих в том числе при расследовании инцидентов ИБ. Тем не менее, Belkasoft Evidence Center и Autopsy не предоставляют возможностей по анализу информации в контексте идентификации, обнаружения и классификации воздействий на файлы, но извлекают данные из массивов (записи журналов событий, BO, pf файлы и др.), которые позволяют в ручном режиме идентифицировать и обнаружить некоторые простые воздействия на файлы. LastActivityView позволяет идентифицировать лишь часть воздействий на файлы, но не обладает функционалом по их обнаружению и классификации.

Возникает потребность в создании комплекса ПС, реализующего полный цикл идентификации, обнаружения и классификации воздействий на файлы при расследовании инцидентов ИБ.

### 1.7. Постановка задач исследования

В главе рассмотрено назначение расследования инцидента ИБ, заключающееся в ликвидации последствий инцидента и выработке рекомендаций по совершенствованию мер ИБ. Проведенный анализ массивов данных, содержащих информацию о воздействиях на файлы, в т.ч. направленных на нарушение ИБ, показал, что не все массивы по отдельности могут быть использованы для идентификации, обнаружения и классификации воздействий на файлы ввиду отсутствия в них полного набора характерных для файла признаков. С целью формализации процесса идентификации воздействий на файлы введено понятие вектора признаков, характеризующих файл, даны определения состояния файла, файловой операции и воздействия на файл. Указанные определения в дальнейшем применяются по тексту работы. Выбран способ формализации процесса идентификации на основе *N*-схем. Обзор существующих методов анализа данных выявил следующие недостатки: перспективный с точки зрения

сокращения объема анализируемых данных кластерный анализ требует точного задания параметров кластеризации, в том числе в ручном режиме, а существующие методы автоматического подбора параметров не всегда дают корректный результат; использование всех значений из записей массива данных в процессе классификации воздействий приводит к снижению точности классификации, а введение новых классов требует проведения ресурсоемких процедур переобучения классификатора, что неприемлемо в рамках обнаружения и классификации воздействий на файлы ввиду отсутствия единого набора классов воздействий во всех существующих КС.

Таким образом, формализация процесса идентификации воздействий на файлы и разработка автоматизированных методов анализа массивов данных, содержащих информацию о воздействиях на файлы, является актуальной задачей. Целью диссертации является разработка автоматизированных методов анализа массивов данных, применяемых при расследовании инцидентов ИБ, а также их программных реализаций и методик использования.

Для достижения поставленной цели требуется решить следующие задачи:

1. Разработка модели процесса идентификации воздействий на файлы с использованием математического аппарата сетей Петри.

2. Разработка математических методов для анализа воздействий на файлы: кластеризационного метода идентификации воздействий и метода экспресс-анализа событий ИБ, свободных от недостатков существующих методов анализа данных, ориентированных на неизменяющиеся во времени наборы признаков.

3. Создание комплекса программных средств, автоматизирующего деятельность специалиста в рамках расследования инцидентов ИБ.

# Глава 2. Разработка математической модели процесса идентификации воздействий на файлы, кластеризационного метода идентификации воздействий и метода экспресс-анализа событий информационной безопасности

Как упоминалось ранее, не все массивы данных содержат полный набор необходимых признаков, характеризующих файл. Для решения задачи идентификации воздействий следует анализировать совокупность массивов, получая из каждого требуемую информацию. Учитывая возможность внесения искажений злоумышленником в массивы данных, возникает потребность в инструменте, который позволит провести классификацию полученных признаков с одновременной верификацией их значений. Рассмотрим процесс изменения признаков, характеризующих файл, при совершении файловой операции, т.е. при изменении состояния файла.

## 2.1. Событийная модель процесса идентификации воздействий на файлы

### 2.1.1. Изменение признаков, характеризующих файл, при совершении файловой операции

Рассмотрим зависимости между файловыми операциями и изменяющимися компонентами вектора $V_f$. Определив изменения компонентов вектора $V_f$, специалист, проводящий расследование инцидента ИБ, может классифицировать ФО в соответствии с таблице 2.1.

Таблица 2.1. Зависимости между ФО и изменяющимися компонентами $V_f$

| **Файловая операция** | **Изменяющиеся компоненты вектора $V_f$** | **Примечание** |
|---|---|---|
| Создание файла ($O1$) | $I_f$, $D_f$, $N_f$, $X_f$ | |
| Удаление файла ($O2$) | $I_f$, $D_f$, $N_f$, $X_f$ | |
| Переименование файла ($O3$) | $N_f$ | |

| **Файловая операция** | **Изменяющиеся компоненты вектора $V_f$** | **Примечание** |
|---|---|---|
| Изменение содержимого файла ($O4$) | $C_f$, $X_f$ | Изменение размера файла повлечет за собой изменение служебной информации о файле |
| Изменение служебной информации ($O5$) | $X_f$ | |
| Перемещение файла в пределах тома ($O6$) | $D_f$ | |
| Копирование файла в пределах каталога ($O7$) | 1) $I_f$, $D_f$, $N_f$, $X_f$ (Создание конечного файла);<br>2) $C_f$, $X_f$ (Изменение содержимого конечного файла, изменение служебной информации конечного файла);<br>3) $X_f$ (Изменение служебной информации исходного файла) | У исходного и конечного файла совпадут $D_f$, но будут отличаться $N_f$ |
| Копирование файла в другой каталог ($O8$) | 1) $I_f$, $D_f$, $N_f$, $X_f$ (Создание конечного файла);<br>2) $C_f$, $X_f$ (Изменение содержимого конечного файла, изменение служебной информации конечного файла);<br>3) $X_f$ (Изменение служебной информации исходного файла) | У исходного и конечного файла совпадут $N_f$, но будут отличаться $D_f$ |

Примечание: исследование зависимостей проводилось в ОС Windows 7 Professional SP1, Microsoft Windows 8.1 Professional, Microsoft Windows 10 Professional с ФС NTFS.

К моменту начала расследования инцидента ИБ файловые операции считаются выполненными. Информация о них может быть получена из рассмотренных ранее массивов данных. В связи с тем, что файловые операции в массивах не всегда указывается в явном виде, необходимо провести анализ набора параметров в данных массивов, чтобы классифицировать файловые операции

$O(t) \in \{O1, O2, O3, O4, O5, O6, O7, O8\}$ (см. таблицу 2.1). Для этого следует определить состояния файла в соответствующие моменты времени и воспользоваться кортежем:

$$O(t_1, t_2) = \langle V_f(t_1), V_f(t_2) \rangle, \tag{6}$$

где $V_f(t_1)$ и $V_f(t_2)$ – векторы признаков, характеризующих файл *f* для начального и конечного состояния соответственно.

Запишем и будем в дальнейшем использовать выражение, описывающее ФО в удобной для восприятия форме:

$$O = \langle V_{f1}, V_{f2} \rangle, \tag{7}$$

где $V_{f1}$ и $V_{f2}$ – векторы признаков, характеризующих файл *f* для начального и конечного состояния соответственно.

При совершении множества файловых операций по отношению к файлу *f* происходит несколько переходов между состояниями $\{V_{f1}, V_{f2}, ..., V_{fm*}, ..., V_{fm}\}$, причем при каждом переходе состояние *m** считается конечным, а *m**-1 – начальным.

Исходя из определений состояния файла и файловой операции, переходы между состояниями файла *f* зависят от примененного к нему набора файловых операций $\{O_1, O_2, ..., O_{m'}, ..., O_{m-1}\}, O_{m'} \in \{O1, O2, O3, O4, O5, O6, O7, O8\}, m' = \overline{1, m-1}$ и сопровождаются изменением значений компонент вектора $V_f$ признаков, характеризующих файл *f*.

Формализовав определение файловой операции опишем воздействие на файл $Y_i$, с учетом выражения (7), следующим образом:

$$Y_i = \left\langle \left\langle O_{fm'} \right\rangle_{m'=1}^{m-1} \right\rangle_{f=1}^{l} = \left\langle \left\langle \left\langle V_{fm'}, V_{f(m'+1)} \right\rangle \right\rangle_{m'=1}^{m-1} \right\rangle_{f=1}^{l}, \tag{8}$$

где $O_{fm'}$ – файловая операция, осуществляемая в отношении файла *f*, $V_{fm'}$ – вектор признаков, характеризующих файл *f* до момента совершения файловой операции $O_{fm'}$ (начальное состояние файла *f*), $V_{f(m'+1)}$ – вектор признаков,

характеризующих файл *f* после совершения файловой операции $O_{fm'}$ (конечное состояние файла *f*), *l* – количество файлов, *m* – количество операций.

Воздействие на файл может представлять собой одну операцию ($f = 1$, $m' = 1$), совершаемую по отношению к файлу. В таком случае оно является простым. Если имеет место последовательность операций и/или набор файлов, то такое воздействие является сложным комплексным ($f \geq 1$, $m' > 1$). Примерами простых воздействий на файлы являются:

- создание, удаление файла;
- изменение имени файла;
- изменение содержимого документа формата txt в текстовом редакторе «Блокнот».

Примеры сложных комплексных воздействий на файлы:

- изменение содержимого документа формата doc, docx, odt в текстовых процессорах Microsoft Word, LibreOffice Writer;
- извлечение файлов из архива с помощью программ-архиваторов;
- шифрование файлов, в т.ч. вредоносным программным обеспечением (Ransomware).

Для построения модели процесса идентификации воздействий на файлы, в первую очередь, необходимо формализовать процесс изменения значений компонентов вектора $V_f$ для классификации файловой операции, совершаемой в отношении файла *f*. Для этого предлагается использовать разработанный алгоритм классификации файловых операций.

### 2.1.2. Алгоритм классификации файловых операций

Алгоритм может быть использован как для классификации одной файловой операции (при отсутствии в массиве данных информации о файловой операции), так и для верификации данных массива (в целях обеспечения достоверности). Входными данными алгоритма являются значения компонентов векторов $V_{f1} = \{ I_{f1}, D_{f1}, N_{f1}, C_{f1}, X_{f1} \}$ и $V_{f2} = \{ I_{f2}, D_{f2}, N_{f2}, C_{f2}, X_{f2} \}$, соответствующих начальному и конечному состояниям файла *f*.

Алгоритм реализуется выполнением следующей последовательности действий:

1. Подать на вход алгоритма значения компонентов векторов $V_{f1}$ и $V_{f2}$.

2. Определить принадлежность $I_{f1}$ и $I_{f2}$ пустому множеству.

3. Если $I_{f1} \in \varnothing$ и $I_{f2} \notin \varnothing$, то классифицируется операция «Создание файла» (таблица 2.1, *O*1).

4. Если $I_{f1} \notin \varnothing$ и $I_{f2} \in \varnothing$, то классифицируется операция «Удаление файла» (таблица 2.1, *O*2).

5. Если $I_{f1} \notin \varnothing$ и $I_{f2} \notin \varnothing$, то необходимо провести дальнейшее сравнение признаков, характеризующих файл.

6. При несовпадении $I_{f1}$ и $I_{f2}$ определяется одна из двух операций копирования (таблица 2.1, *O*7 и *O*8) с учетом равенства $D_{f1}, D_{f2}$ и $N_{f1}, N_{f2}$.

7. В случае совпадения $I_{f1}$ и $I_{f2}$ определяется равенство $N_{f1}$ и $N_{f2}$, а также равенство $D_{f1}$ и $D_{f2}$.

8. Если не равны имена файла, но равны идентификаторы его родительского каталога, то классифицируется операция «Переименование файла» (таблица 2.1, *O*3).

9. Если равны имена файла, но не равны идентификаторы его родительского каталога, то классифицируется операция «Перемещение файла в пределах логического раздела» (таблица 2.1, *O*6).

10. В случае совпадения имен файла и идентификаторов его родительского каталога необходимо сравнить $C_{f1}$ и $C_{f2}$.

11. Если $C_{f1}$ и $C_{f2}$ равны, то классифицируется операция «Изменение служебной информации» (таблица 2.1, *O*5), в противном случае — «Изменение содержимого файла» (таблица 2.1, *O*4).

Результатом работы алгоритма является классификация файловой операции. Стоит отметить, что элемент «Вызов исключения» является условным и указывает на ошибку в значениях компонентов векторов $V_{f1}$ и/или $V_{f2}$.

Блок-схема алгоритма классификации файловых операций представлена на рисунке 2.1.

При применении алгоритма следует учитывать ряд особенностей:

1. $\neg\exists V_f \mid I_f \notin \varnothing, D_f \in \varnothing$ (файл $f$ всегда расположен в его родительском каталоге).

2. $N_{f1}, N_{f2} \notin \varnothing$, так как файл не может существовать без имени.

3. Из рисунка 2.1 видно, что для классификации всех файловых операций отсутствует необходимость сравнения значений компонентов $X_{f1}$ и $X_{f2}$ ввиду отсутствия неопределенности в работе алгоритма. В связи с этим, при разработке модели компоненты $X_f$ и $C_f$ заменены на признак $Z_f = g(X_f, C_f)$ — признак изменения содержимого файла, который может быть сформирован на основании данных, полученных при анализе служебной информации файла (изменение размера содержимого файла, временных отметок и т.п.) и его содержимого. $Z_f$ принимает два значения — 1, если изменилось содержимое файла, и 0, если изменилась его служебная информация.

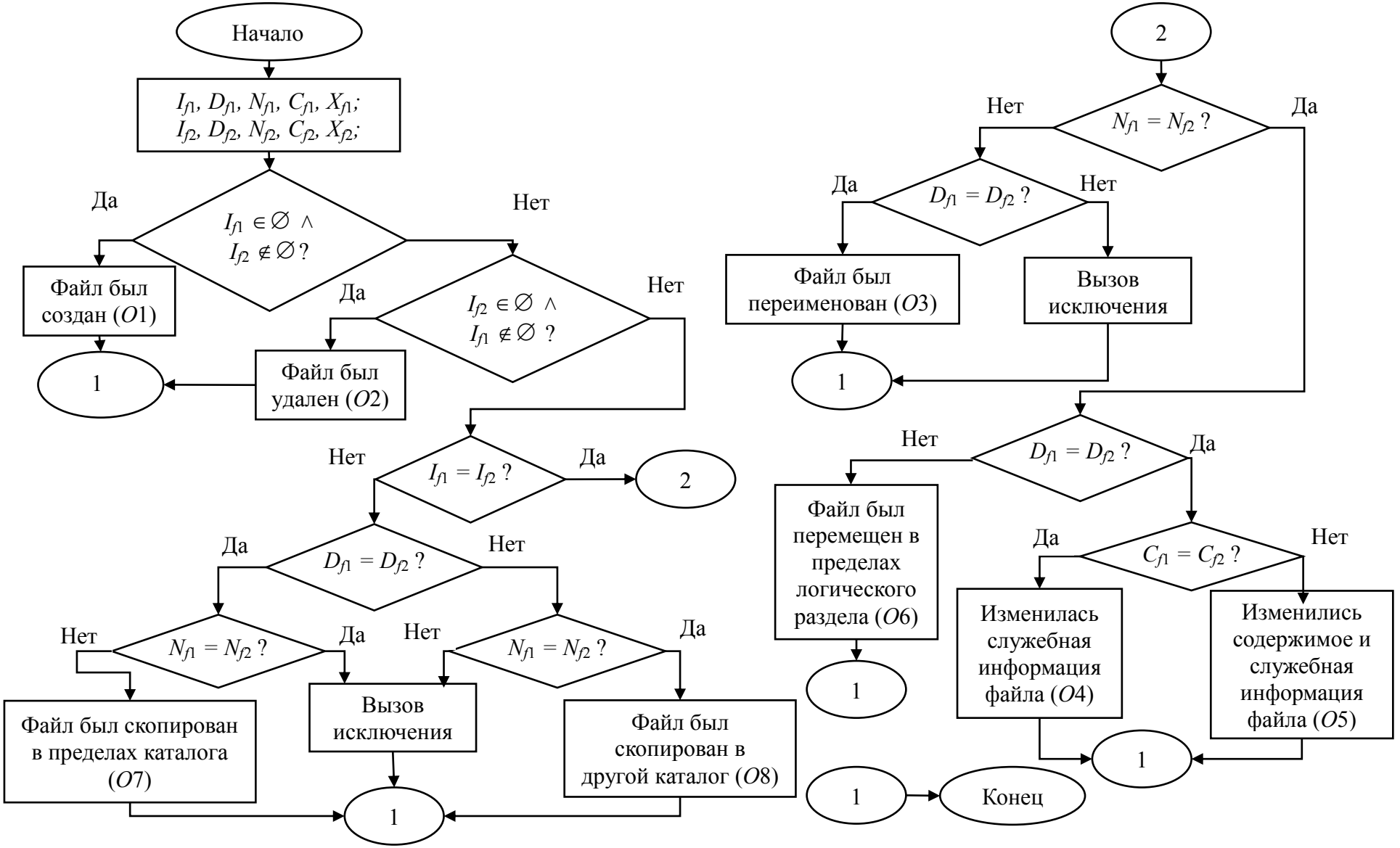


Рисунок 2.1. Блок-схема алгоритма классификации файловых операций

Представленный алгоритм не учитывает сложную природу компьютерной системы, в которой выполняются различные по типу и назначению процессы: последовательные и параллельные, синхронные и асинхронные. Среди множества процессов встречаются такие, которые связаны с воздействиями на файлы, результат работы которых необходимо фиксировать. Как уже было сказано ранее, сложные комплексные воздействия на файлы состоят более чем из одной файловой операции и могут иметь отношение к нескольким файлам. Для моделирования процесса идентификации как простых, так и сложных воздействий на файлы, а не только файловых операций, необходим математический аппарат, позволяющий учесть множество возможных комбинаций состояний файлов. При описании систем подобного типа используются *N*-схемы в виде сети Петри [134], которые позволяют смоделировать процесс идентификации воздействий на файлы, формализовав алгоритм, изображенный на рисунке 2.1 с учетом выражений (6) и (8).

### 2.1.3. Применение математического аппарата сетей Петри для моделирования процесса идентификации воздействий на файлы

Рассматриваемая модель процесса идентификации воздействий на файлы, построенная с использованием сети Петри, дополняется двумя параметрами: $\delta$, который является внешним по отношению к сети «накопителем» файловых операций, и временной задержкой $\tau = t_2 - t_1$, где $t_1$ соответствует времени до начала файловой операции, а $t_2$ – времени ее окончания. Таким образом, $Net = (P, T, E, Q, \mu, \delta, \tau)$. Предложенная далее сеть Петри *Net* построена таким образом, чтобы исключить случайные переходы.

Представленная сеть *Net* характеризуется следующими особенностями (назначение позиций и переходов указано далее):

1. В конечное множество позиций $P = P_{fs} \cup P_{al} \cup P_{fin} \cup P_{add}$ включены:

• позиции состояний файла $P_{fs}$ ($P_1 \in \langle I_{f1}, D_{f1}, N_{f1} \rangle$, $P_2 \in \langle I_{f2}, D_{f2}, N_{f2} \rangle$, $P_6 \in Z_f$);

• позиции состояний алгоритма $P_{al}$ ($P_3$ — $P_5$, $P_{19}$ — $P_{28}$), представленного на рисунке 2.1, (с учетом признака $Z_f$);

• позиции классификации файловых операций $P_{fin}$ ($P_{29}$ — $P_{36}$);

• вспомогательные позиции, предназначенные для корректного функционирования сети Петри $P_{add}$ ($P_7$ — $P_{18}$, $P_{37}$, $P_{38}$).

2. В конечное множество переходов $T = T_{fin} \cup T_{al} \cup T_{add} \cup T_{delay} \cup T_{fo}$ включены:

• переходы «индикации» получения значений состояний файла $T_{fin}$ ($T_1$, $T_2$);

• переходы алгоритма $T_{al}$ ($T_3$ — $T_8$, $T_{13}$ — $T_{16}$), представленного на рисунке 2.1 (с учетом признака $Z_f$);

• вспомогательные переходы, предназначенные для корректного функционирования сети Петри $T_{add}$ ($T_9$ — $T_{12}$);

• переход имитации временной задержки при совершении файловой операции $T_{delay}$ ($T_{25} \in \tau : \tau = t_2 - t_1$);

• переходы классификации файловых операций $T_{fo}$ ($T_{17} \in O1, T_{18} \in O2, T_{19} \in O3, T_{20} \in O5, T_{21} \in O4, T_{22} \in O6, T_{23} \in O7, T_{24} \in O8$) (типы файловых операций представлены в таблице 2.1).

В рамках модели все переходы, кроме $T_{25}$, являются примитивными, т.е. выполняются мгновенно.

3. Входные и выходные функции – E и Q соответственно ($E: T_j \rightarrow P_j$, $Q: P_j \rightarrow T_j$), отображающие позиции в переходы и наоборот, связаны как с алгоритмическими, так и вспомогательными позициями и переходами.

4. Маркировка сети фишками $\mu : \|\mu\| = \|P\|, \mu(P_6) \in Z_j$.

5. Конечное множество значений временных задержек нетривиальных переходов τ связано с переходом $T_{25}$.

Начальное состояние сети Петри, обозначаемое как $\mu$, маркируется фишками только для позиций $P_1$ — $P_6$ и $P_{37}$, в остальных позициях отсутствие фишек является обязательным условием корректности работы сети. Маркировка позиций $P_1$ — $P_6$ осуществляется внешним по отношению к моделируемому процессу элементом. Позиции $P_7$ — $P_{28}$ и переходы $T_3$ — $T_{24}$ реализуют ветвления алгоритма, представленного на рисунке 2.1. По достижению одной из позиций $P_{29}$ — $P_{36}$ классифицируется файловая операция в рамках моделируемого процесса, с учетом данных из позиций $P_1$ и/или $P_2$. Позиции $P_{29}$ — $P_{36}$ взаимоисключающие. Полученный результат помещается в «накопитель» $\delta$, который содержит все операции в рамках одного воздействия на файлы. Принудительное прерывание выполнения сети обеспечивается удалением фишки из позиции $P_{37}$ в момент подачи на вход сети новых данных о состояниях файла. Прерывание сигнализирует о завершении «накопления» файловых операций, т.е. об окончании идентификации воздействия на файл в соответствии с выражением (8). Результатом выполнения сети является идентифицированное воздействие на файл.

Стоит отметить, что при неверной совокупности заданных фишек в позициях $P_1$ — $P_6$ и $P_{37}$ выполнение сети прервется раньше, чем будут достигнуты $P_{29}$ — $P_{36}$. Такие ситуации должны быть рассмотрены специалистом с позиции достоверности заданных значений параметров, а также с точки зрения осуществления файловых операций с другим файлом в рамках одного сложного комплексного воздействия. Конечное множество маркировок, обеспечивающих определение файловой операции, представлено в таблице 2.2.

Таблица 2.2. Множество маркировок позиций

| **Маркировка μ** | **Файловая операция** |
|---|---|
| $P_1 = 0$, $P_2 = 1$, $P_3 = 0$, $P_4 = 0$, $P_5 = 0$, $P_6 = 0$ | Создание файла (таблица 2.1, $O1$) |
| $P_1 = 1$, $P_2 = 0$, $P_3 = 0$, $P_4 = 0$, $P_5 = 0$, $P_6 = 0$ | Удаление файла (таблица 2.1, $O2$) |

| Маркировка μ | Файловая операция |
|---|---|
| $P_1 = 1$, $P_2 = 1$, $P_3 = 1$, $P_4 = 1$, $P_5 = 0$, $P_6 = 0$ | Переименование файла (таблица 2.1, *O*3) |
| $P_1 = 1$, $P_2 = 1$, $P_3 = 1$, $P_4 = 1$, $P_5 = 1$, $P_6 = 1$ | Изменение содержимого файла (таблица 2.1, *O*4) |
| $P_1 = 1$, $P_2 = 1$, $P_3 = 1$, $P_4 = 1$, $P_5 = 1$, $P_6 = 0$ | Изменение служебной информации файла (таблица 2.1, *O*5) |
| $P_1 = 1$, $P_2 = 1$, $P_3 = 1$, $P_4 = 0$, $P_5 = 1$, $P_6 = 0$ | Перемещение файла в пределах логического раздела (таблица 2.1, *O*6) |
| $P_1 = 1$, $P_2 = 1$, $P_3 = 0$, $P_4 = 1$, $P_5 = 0$, $P_6 = 0$ | Копирование файла в пределах каталога (таблица 2.1, *O*7) |
| $P_1 = 1$, $P_2 = 1$, $P_3 = 0$, $P_4 = 0$, $P_5 = 1$, $P_6 = 0$ | Копирование файла в другой каталог (таблица 2.1, *O*8) |

Сеть Петри, описывающая разработанную модель, изображена на рисунке 2.2. Описание позиций и переходов представлено в таблице Б.1 и таблице Б.2 приложения Б соответственно.

В результате анализа конечного состояния предложенной сети Петри был выявлен ряд особенностей:

1. Операции «Создание файла» и «Удаление файла» определяются независимо от маркировок позиций $P_3$ — $P_6$.

2. Операции «Изменение служебной информации файла» и «Изменение содержимого файла» определяются только при условиях: $I_{f1} = I_{f2}$, $D_{f1} = D_{f2}$, $N_{f1} = N_{f2}$.

3. Операции, связанные с копированием файла, являются составными и предполагают как создание служебной информации о новом файле, так и копирование содержимого (при его наличии) исходного файла согласно таблице 2.1. Они представлены отдельными позициями сети, не зависящими от операций «Создание файла» и «Изменение содержимого файла».

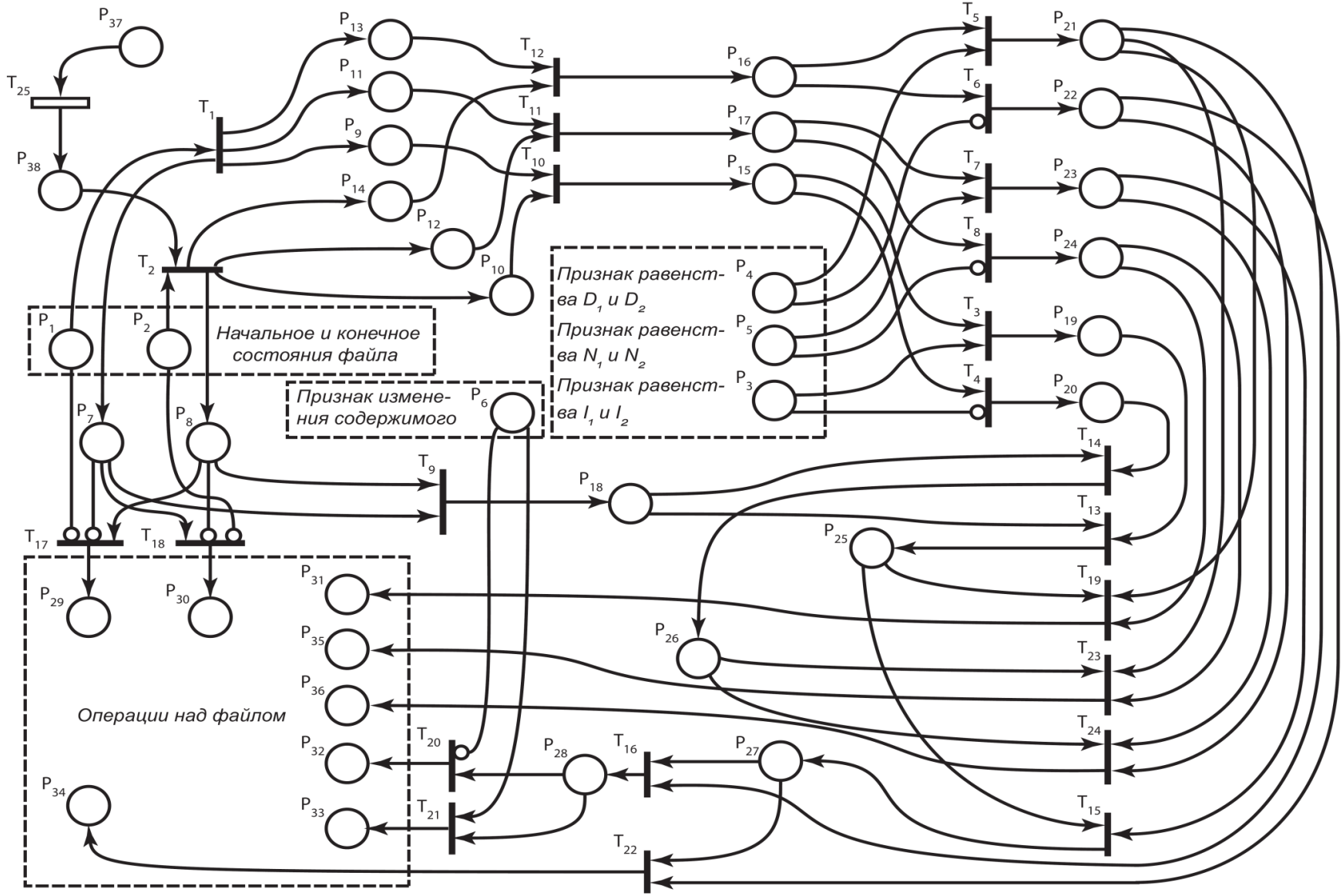


Рисунок 2.2. Сеть Петри, описывающая модель процесса идентификации воздействий на файлы

Предложенная модель позволяет идентифицировать воздействие на файл на основании данных из вышеуказанных массивов. Как упоминалось ранее, не все массивы содержат полный набор компонентов, из которых можно сформировать вектор $V_f$, поэтому возможно извлечение данных из совокупности массивов. После извлечения специалист проводит оценку равенства компонент $I_f$, $D_f$, $N_f$, определяет признак $Z_f$ и получает результат в виде классифицированной файловой операции, которая помещается в накопитель $\delta$. Модель также позволяет провести верификацию массивов данных – последовательную классификацию файловых операций на основе извлекаемых признаков, характеризующих файл, до выявления случая, когда сеть не завершила свою работу, что будет свидетельствовать о некорректных значениях данных в массиве.

Рассмотрим выполнение предложенной сети на примере идентификации простого воздействия на файл *f*. Пусть начальное состояние файла *f* описывается вектором $V_{f1} = \{ I_{f1}, D_{f1}, N_{f1}, C_{f1}, X_{f1} \}$, а конечное — $V_{f2} = \{ I_{f2}, D_{f2}, N_{f2}, C_{f2}, X_{f2} \}$, как показано в таблице 2.3.

Таблица 2.3. Пример процесса идентификации воздействия на файл *f*

| **Признаки, характеризующие файл** | **Начальное состояние, описываемое вектором $V_{f1}$** | **Конечное состояние, описываемое вектором $V_{f2}$** |
|---|---|---|
| $I_f$ | 12876 | 12876 |
| $D_f$ | 13492 | 13486 |
| $N_f$ | Напоминание.txt | Напоминание.txt |
| $X_f$ | ⟨1,4,130456987653⟩ | ⟨1,4,130456987653⟩ |
| $C_f$ | Встреча в 10.00 | Встреча в 10.00 |

Из таблицы 2.3 видно, что $I_{f1} = I_{f2}$, $D_{f1} \neq D_{f2}$, $N_{f1} = N_{f2}$, $C_{f1} = C_{f2}$, $X_{f1} = X_{f2}$. Укажем маркировку $\mu$ – входные данные предложенной сети. В позиции $P_1$ и $P_2$ фишки устанавливаются, т.к. файл существует в обоих состояниях. Исходя из равенства $I_f$ и $N_f$, следует установить фишки в позиции $P_3$ и $P_5$. Учитывая

равенство $C_f$ и $X_f$ и неравенство $D_f$, не следует устанавливать фишки в позиции сети $P_4$ и $P_6$. Запуск сети производится после установки фишки в позицию $P_{37}$. Выполнение сети заключается в последовательном переходе фишек по позициям (пример перемещения фишек по позициям указан на рисунках Д.1-Д.10 приложения Д). Результатом выполнения сети является классифицированная файловая операция «Перемещение», т.к. фишка оказалась в позиции $P_{34}$. Классифицированная операция помещается в накопитель $\delta$. В связи с тем, что в рассматриваемом примере рассматривалось простое воздействие, состоящее из одной файловой операции, фишку в позицию $P_{37}$ помещать не требуется, и выполнение сети прервется.

Аналогичным образом могут быть классифицированы остальные воздействия при определении всех состояний интересующих файлов. Представленная сеть смоделирована и протестирована в специализированном ПО Platform Independent Petri Net Editor [135, 136]. Соответствие результата, полученного после выполнения сети, реальным файловым операциям экспериментально подтверждено на специальном стенде с установленными ОС Microsoft Windows 7 Professional SP1, Microsoft Windows 8.1 Professional, Microsoft Windows 10 Professional. Вместе с тем, способ моделирования процесса идентификации воздействий на файлы с использованием математического аппарата сетей Петри является универсальным и может быть применен в других ОС и ФС с учетом имеющихся в них признаков, характеризующих файлы.

## 2.2. Кластеризационный метод идентификации воздействий на файлы

В тех ситуациях, когда объем данных в массиве, содержащем информацию о воздействиях на файлы, измеряется десятками-сотнями записей, возможен его ручной анализ. Тем не менее, анализ «вручную» обладает рядом недостатков:

1. Человеческий фактор – специалист, проводящий анализ массивов данных может упустить из виду информацию, связанную с произошедшим инцидентом;

2. Начало инцидента могло произойти значительно раньше, чем основная совокупность воздействий на файлы, обрабатываемые в КС, в результате чего потенциальный «источник инцидента» не попадет в выборку по временному интервалу.

Для идентификации воздействий на файлы возможно использование журнала изменений тома $UsnJrnl в качестве массива данных, который обладает рядом преимуществ:

1. Полнота — журнал содержит подробную информацию о действиях, совершенных по отношению к файлам;

2. Достоверность — системный файл $UsnJrnl защищен ОС от несанкционированных изменений со стороны пользователя;

3. Простота обработки данных, что играет немаловажную роль в последующей автоматизации их анализа.

Рассмотрим структуру кластеризационного метода идентификации воздействий на файлы основанного на использовании алгоритма *k*-средних при обработке записей журнала изменений тома $UsnJrnl.

### 2.2.1. Анализ зависимости в формировании записей журнала изменений тома $UsnJrnl

Исследование порядка формирования записей позволило выявить зависимость, описываемую выражением:

$$\Psi = g(\tau, K_f) \tag{9}$$

где $\Psi$ – значение вектор-функции $g$, содержащее совокупность номеров записей журнала изменений тома $UsnJrnl, имеющих отношение к файлу $f$, причем $\forall f\ U_f \in \Psi$ (см. $U_f$ в таблице 1.1); $K_f$ – количество записей журнала, имеющих отношение к файлу $f$; $\tau$ – множество временных интервалов различной величины, в рамках которых осуществляются

действия по отношению к файлу $f$, описываемые записями журнала изменений тома \$UsnJrnl, причем $T_f \in \tau$ (см. $T_f$ в таблице 1.1).

Таким образом, записи журнала изменений тома \$UsnJrnl можно представить в виде точек на плоскости для последующего анализа. Такое представление является основной идеей рассматриваемого кластеризационного метода идентификации воздействий на файлы, которая будет прослеживаться в составных частях метода – алгоритмах №№1-3, рассмотренных далее.

Пример зависимости (9), аппроксимированной кусочно-заданной функцией, представлен на рисунке 2.3. По оси абсцисс отложены отнормированные к минимальному значения $T_f$ (обозначены как $T/\min(T)$), а по оси ординат – отнормированные к минимальному значения $U_f$ (обозначены как $U/\min(U)$) нескольких записей журнала \$UsnJrnl. Окружностями обозначены значения ($T_f$; $U_f$) каждой записи для файла $f$.

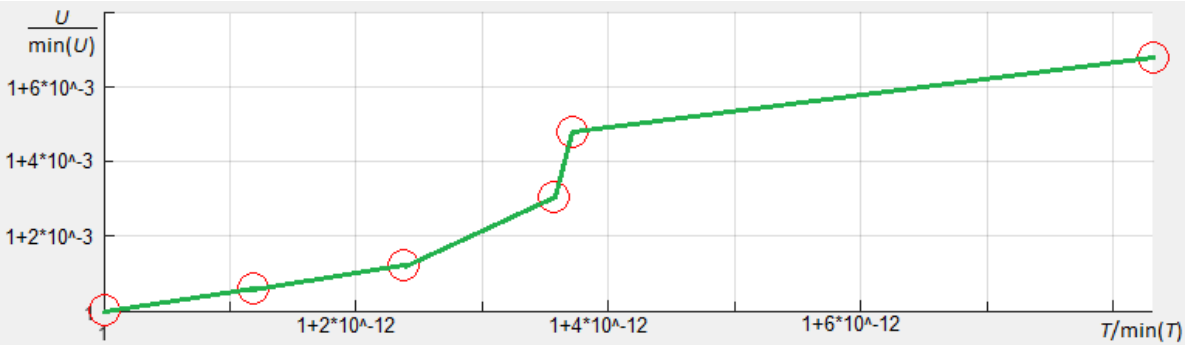


Рисунок 2.3. Зависимость отнормированных к минимальному значений $U_f$ от количества записей журнала \$UsnJrnl и отнормированных к минимальной ВО появления записей $T_f$ в нескольких временных интервалах

Выявленная зависимость обладает следующими свойствами:

- нелинейная – за фиксированный интервал времени может быть совершено произвольное количество воздействий на файлы, что повлечет за собой появление соответствующего количества записей журнала \$UsnJrnl;
- возрастающая – номер записи постоянно увеличивается. Несмотря на наличие ограничения в размерности поля $U_f$ (8 байт), исчерпать его ресурс при повседневной активности пользователя затруднительно.

Нелинейность позволяет определить период наибольшей активности пользователя/процесса по отношению к файлу. Крутизна графика функции в выбранном фиксированном временном интервале характеризует интенсивность воздействий на файлы. Рост значения номера записей $U_f$ говорит об интенсивности работы с файлами в целом. При совершении простых воздействий крутизна графика функции незначительна. К ним можно отнести работу с документами в формате txt, простые файловые операции, такие как создание/удаление/переименование файла. Сложные комплексные воздействия на файлы существенно увеличивают крутизну графика. Примерами могут служить редактирование «офисного» документа в текстовом процессоре Microsoft Word, разархивирование файлов и т. д. Изменяемое значение крутизны графика и нелинейность позволяют рассмотреть использование математических алгоритмов и методов машинного обучения для проведения анализа выявленной зависимости.

### 2.2.2. Алгоритм № 1 кластеризационного метода для разделения записей журнала $UsnJrnl на блоки

Объем записей в журнале $UsnJrnl может превышать 600 тысяч, что, в свою очередь, затрудняет использование алгоритма *k*-средних для группировки записей в рамках предлагаемого метода идентификации воздействий на файлы при расследовании инцидентов ИБ – продолжительность анализа записей может достигать нескольких часов в зависимости от аппаратной конфигурации КС. Для повышения скорости работы алгоритма *k*-средних в отношении решаемой задачи необходимо подготовить входные данные.

Для повышения скорости анализа предлагается использовать в качестве входных данных алгоритма *k*-средних блоки записей из $UsnJrnl, предварительно разделенных согласно идентификаторам файлов $I_f$. Помимо выборки данных по $I_f$ необходимо также учесть особенность работы драйвера ФС NTFS, а именно активное использование им существующих освобожденных файловых записей вместо выделения новых. Указанная

особенность может привести к получению неверного результата.

Для предотвращения ошибок анализа записей журнала $UsnJrnl из-за особенностей работы драйвера ФС NTFS при разделении записей по идентификаторам файлов $I_f$ предлагается использовать следующий алгоритм:

1. Выбрать записи в соответствии с идентификатором $I_f$.
2. Создать буфер «разделенных» записей – буфер 1 и назначить его текущим.
3. Провести в цикле последовательную проверку выбранных записей на наличие значения $R_f = 512$ (таблица 1.2), свидетельствующего об удалении файла.
4. Если значение 512 в поле $R_f$ записи не найдено, то поместить запись в текущий буфер.
5. В противном случае создать следующий буфер (буфер 2), назначить его текущим. Если значение 512 в поле $R_f$ записей появляется $x$ раз, то создать $x$+1 буферов, чтобы разделить записи на несколько частей в соответствии с особенностями работы драйвера ФС NTFS.
6. Поместить запись в текущий буфер.
7. Разделенные на части записи сохранить в виде блоков $b$ ($b \in B$) для последующего анализа.

Блок-схема алгоритма представлена на рисунке 2.4.

Результатом работы алгоритма является сформированное множество блоков $B$ записей журнала $UsnJrnl с одинаковым значением $I_f$.

Графическое пояснение результатов работы алгоритма представлено на рисунке 2.5.

Говоря об оценке вычислительной сложности алгоритма следует учитывать, что наибольший вклад вносит процедура выборки записей журнала $UsnJrnl, вычислительная сложность которой за счёт использования двоичных деревьев поиска в индексах базы данных, хранящей записи загруженного журнала, составляет $O(n\log_2 n)$, где $n$ – количество записей журнала.

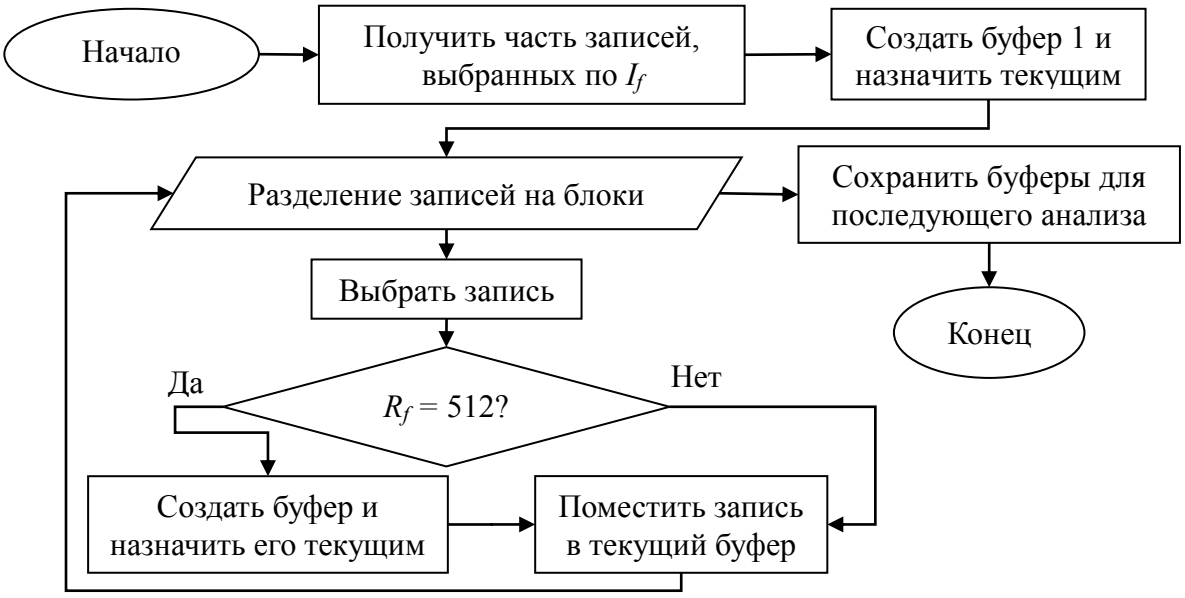


Рисунок 2.4. Блок-схема алгоритма разделения записей на блоки

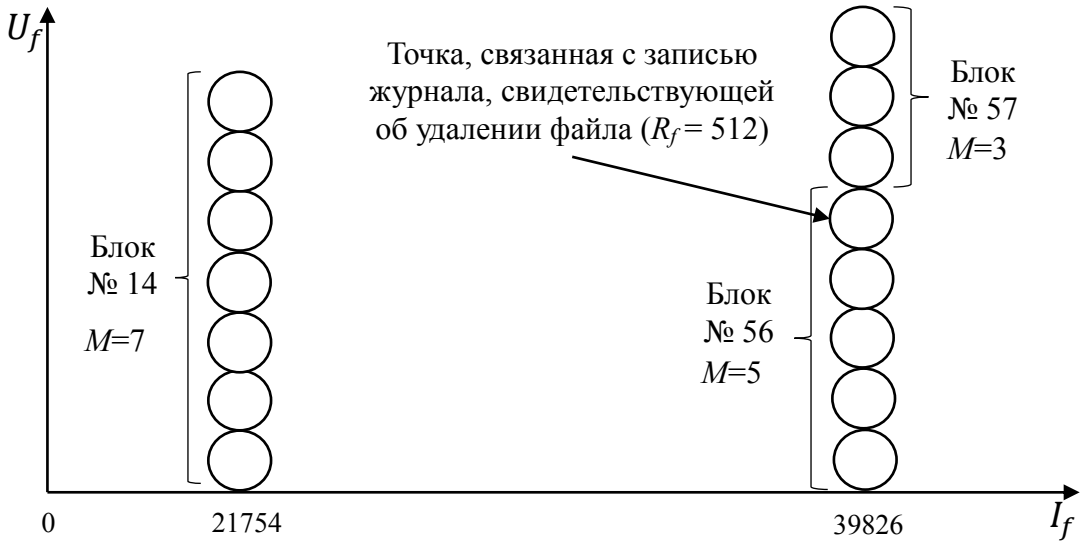


Рисунок 2.5. Графическое пояснение назначения алгоритма

Перед применением алгоритма № 2 к полученным блокам записей необходимо применить два преобразования. Во-первых, записям в рамках блока следует присвоить порядковые номера. Таким образом, значения поля «Номер записи» представляются в виде $U_f = \{U_{m,f} \mid m = \overline{1,M}\}$, а «Временная отметка создания записей» – $T_f = \{T_{m,f} \mid m = \overline{1,M}\}$, где $m$ – порядковый номер записи журнала в блоке, а $M$ – количество записей журнала в блоке. Во-вторых, значения $T_{m,f}$ и $U_{m,f}$ необходимо отнормировать к своим

минимумам: $T'_{m,f} = \frac{T_{m,f}}{\min(T_{m,f})}$, $U'_{m,f} = \frac{U_{m,f}}{\min(U_{m,f})}$. Предложенная нормировка позволит работать с точками в пространстве в числовых значениях одного порядка, чтобы скопления точек не были «растянуты» по осям во избежание получения некорректных результатов алгоритмом кластеризации *k*-средних.

### 2.2.3. Алгоритм № 2 кластеризационного метода для автоматического определения оптимального количества кластеров *k* и распределения точек по кластерам

Для того, чтобы воспользоваться формулой (5) для определения оптимального значения *k*, необходимо внести несколько изменений.

Первое изменение формулы (5) связано с функцией правдоподобия. В качестве ее основы может быть выбрана плотность распределения вероятности, которая применима к непрерывным случайным величинам. Точки на плоскости являются дискретными величинами и не могут использоваться в качестве аргумента функции плотности распределения вероятности (PDF). Переход от дискретных величин к непрерывным осуществим следующим образом: каждая точка пространства, ассоциированная с кластером, удалена на некоторое расстояние от его центра. Это расстояние представим в виде вектора. В качестве аргумента PDF будем использовать норму получившихся векторов. Таким образом, в результате осуществленного перехода, становится возможным использование PDF как основы функции правдоподобия. Тогда формула (5) приобретает вид:

$$L(x_m, c) = -\sum_{m=1}^{M} \log_2 p(\|x_m - c\|) + \frac{1}{2} P \log_2 M, \qquad (10)$$

где $x_m$ – точка с координатами $(T'_{m,f}, U'_{m,f})$, $c$ – центр кластера (определенный с помощью алгоритма *k*-средних) с рассчитанными координатами $(T'_c, U'_c)$, с которым $x_m$ ассоциирована, $p(\|x_m - c\|)$ – плотность

распределения вероятности нормы вектора между $x_m$ и $c$, $M$ – количество записей \$UsnJrnl в блоке, т.е. количество анализируемых точек.

Второе изменение формулы (5) связано с учетом используемого количества кластеров для описания набора данных. В работе [137] автор использовал расчет MDL для определения $k$ в алгоритме $k$-медоидов, применив норму вектора от точек до центров кластеров, а также добавив в формулу (5) дополнительное слагаемое, которое «штрафует» рост количества указываемых алгоритму кластеров. Основываясь на полученном в [137] результате, с учетом дополнительного слагаемого, формула (10) приобретает следующий вид:

$$L(x_m, c) = -\sum_{m=1}^{M} \log_2 p(\|x_m - c\|) + \frac{1}{2} P \log_2 M + k \log_2 M, \qquad (11)$$

где $k$ – используемое количество кластеров для группировки точек.

Третье изменение формулы (5) связано со слагаемым, которое определяет «штраф» за увеличение количество оцениваемых параметров модели. При расчете нескольких значений MDL для определения $k = \overline{1, M}$, количество параметров $P$, задающих форму графика PDF, не изменяется. Количество точек в блоке также остается постоянным. Исходя из того, что все компоненты второго слагаемого в рамках расчета зависимости значения MDL от $k$ остаются постоянными, его влияние на график зависимости заключается лишь в смещении по оси ординат. В рамках решаемой задачи не требуется получение абсолютного значения $L(x_m, c)$, в связи с чем расчет второго слагаемого становится излишним. Окончательная формула для расчета MDL в целях определения оптимального значения $k$, описывается выражением:

$$L(x_m, c) = -\sum_{m=1}^{M} \log_2 p(\|x_m - c\|) + k \log_2 M. \qquad (12)$$

Согласно принципу MDL, оптимальное значение $k$ соответствует минимальному значению $L(x_m, c)$, рассчитанному по формуле (12). По

результатам нахождения оптимального значения $k$ к блоку записей журнала \$UsnJrnl применяется выбранный ранее алгоритм кластеризации $k$-средних.

В ходе экспериментов расчет $L(x_m,c)$ проводился для нескольких типов PDF, в отличие от работы [137]. Необходимость выбора из нескольких PDF обусловлена тем, что при расчете $L(x_m,c)$ в рамках предлагаемого метода анализа записей журнала \$UsnJrnl предполагаемое распределение величин расстояний между значениями $x_m$ – точка с координатами ($T'_{m,f}$, $U'_{m,f}$), $c$ – центр кластера с рассчитанными координатами ($T'_c$,$U'_c$), с которым $x_m$ ассоциирована, задается специалистом исходя из его оценки полученных результатов. Влияние на распределение величин расстояний оказывают типы воздействий на файлы, а также интенсивность этих воздействий. Результаты экспериментов по выбору значений параметров PDF обобщены в таблице 2.4. Подробные данные представлены в таблицах приложения В.

Таблица 2.4. Значения параметров PDF, используемых при определении оптимального количества кластеров $k$

| **PDF** | **Параметр** | **Диапазон значений параметра / Оптимальное значение параметра** |
|---|---|---|
| Нормальное распределение | $\mu$ | 0-2 / 0 |
| | $\sigma$ | 0.5-1.5 / 1 |
| Гамма-распределение | $\alpha$ | 0.5;1;2 / 1 |
| | $\beta$ | 0.8-1.1 /0.9 |
| Распределение Стьюдента | $\nu$ | 1-5 / 1 |
| Распределение $\chi^2$ | $\nu$ | 1-5 / 2 |
| Распределение Фишера | $\nu 1$ | 1-2 / 1 |
| | $\nu 2$ | 1-5 / 2 |

В ходе экспериментов было установлено, что достаточно использовать PDF нормального или гамма-распределения с оптимальными значениями параметров, указанными в таблице 2.4, при расчёте значения $L(x_m,c)$ для определения оптимального количества кластеров $k$.

Алгоритм № 2 для автоматического определения оптимального количества кластеров $k$ и распределения точек по ним реализуется выполнением следующей последовательности действий:

1. Выбрать блок $b$ записей журнала \$UsnJrnl, предварительно

обработанных с использованием алгоритма № 1.

2. Выбрать PDF с оптимальными параметрами согласно таблице 2.4.

3. Определить количество записей $M$ в блоке $b$, с которыми будут ассоциированы точки в пространстве.

4. Запустить цикл с изменением значения $k$ от 1 до $M$. На каждой итерации цикла:

   a. с использованием алгоритма $k$-средних для текущего значения $k$ вычислить центры кластеров $c_m$ текущего блока записей;

   b. с использованием формулы (12) рассчитать значение $L(x_m, c)$;

   c. сохранить вычисленное значение $L(x_m, c)$ в накопитель.

5. Определить оптимальное значение $k$, соответствующее минимальному $L(x_m, c)$.

6. Сохранить результаты проведенной кластеризации блока $b$ записей журнала \$UsnJrnl при оптимальном значении $k$.

Графическое пояснение результатов работы алгоритма № 2 представлено на рисунке 2.6.

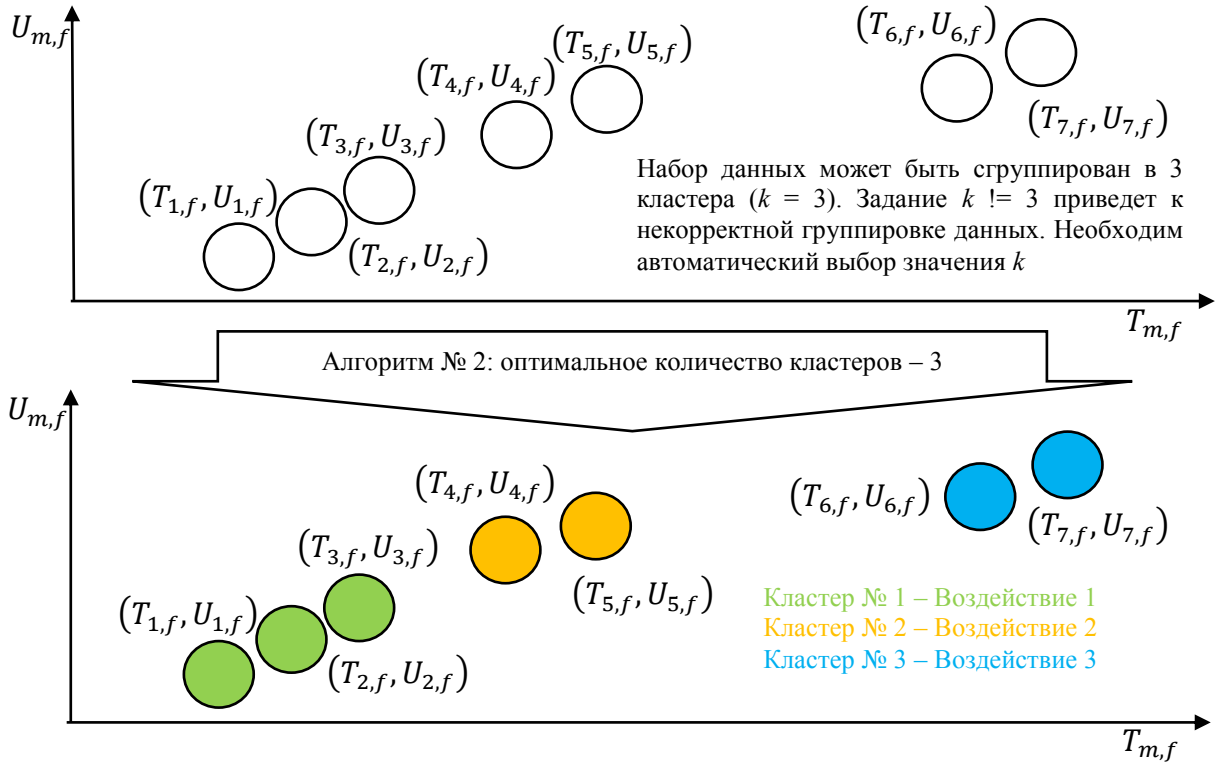


Рисунок 2.6. Группировка точек на плоскости $U_{m,f}(T_{m,f})$ в $k$ = 3 кластера

Пример определения оптимального количества кластеров $k$ и кластеризации записей журнала \$UsnJrnl показан на рисунке 2.7. На верхней оси координат: пунктирной линией показан график значений $L(x_m, c)$ (обозначено как $L(x)$), рассчитанных для нескольких значений $k$ по формуле (11); сплошная линия показывает график значений $L(x_m, c)$, рассчитанных по формуле (12); звездой отмечено оптимальное значение $k$. На нижней оси координат: окружностями обозначены значения $x_m$ – точки с координатами $(T'_{m,f}, U'_{m,f})$, полученные из записей журнала \$UsnJrnl, плюсами – центры кластеров $c$.

Как видно из графиков верхней части рисунка, слагаемое $\frac{1}{2} P \log_2 |X|$ не оказывает влияния на форму графика. В результате проведенной кластеризации блоков записей журнала \$UsnJrnl выделены два воздействия на файл $f$, состоящие из 12 и 4 записей журнала изменений тома \$UsnJrnl соответственно. Центры кластеров, описывающих выделенные воздействия, обозначены плюсами.

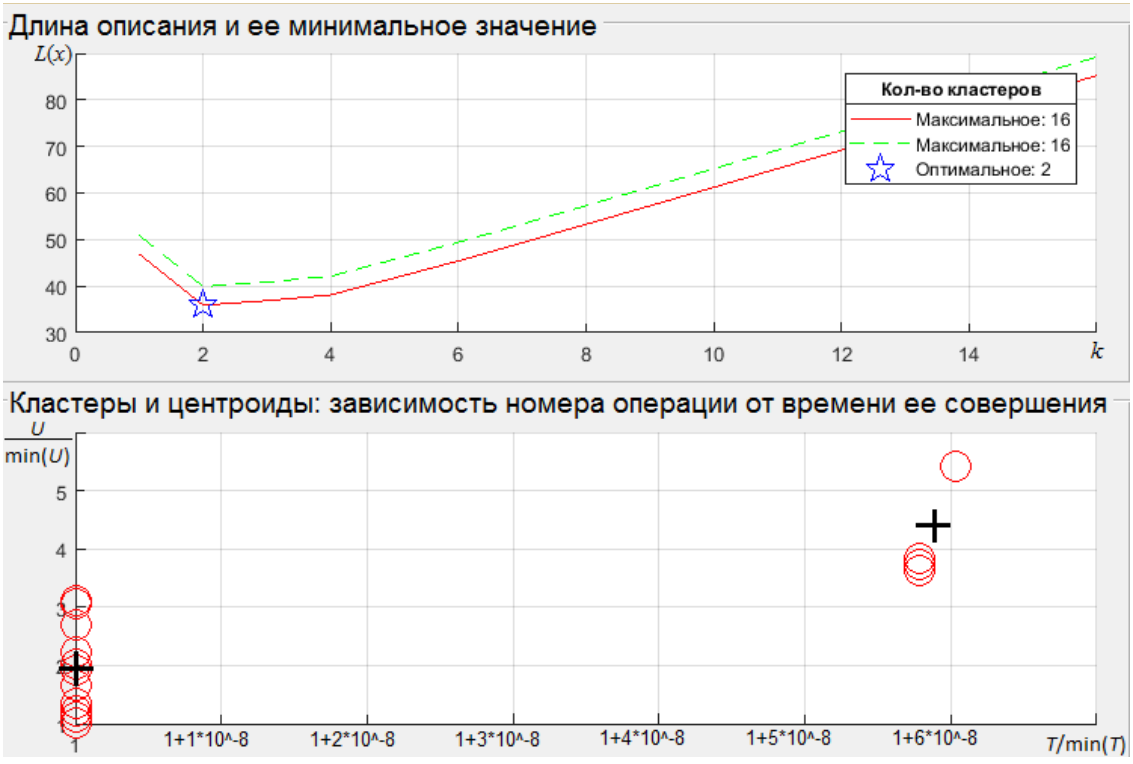


Рисунок 2.7. Пример определения оптимального количества кластеров для идентификации воздействия на файл

При предварительной подготовке данных алгоритмом № 1 было сделано допущение о возможности разделения записей журнала $UsnJrnl по идентификаторам файловых записей. Такое допущение позволило уменьшить время обработки данных с применением алгоритма *k*-средних, но в то же время снизило способность метода к идентификации сложных комплексных воздействий, состоящих из нескольких записей журнала $UsnJrnl, в том числе имеющих отношение к различным файлам. Для того, чтобы корректно идентифицировать сложные комплексные воздействия, необходимо проанализировать связи между полученными кластерами.

Вычислительная сложность алгоритма определяется двумя составляющими: процедура выборки записей журнала $UsnJrnl, вычислительная сложность которой составляет $O(n'\log_2 n')$, где $n'$ – количество записей журнала в блоке и алгоритмом *k*-средних, вычислительная сложность которого, как упоминалось ранее, составляет $O(kid'n')$, где $k$ – количество кластеров, $i$ – количество итераций алгоритма, $d'$ – размерность вектора входных данных, $n'$ – количество точек. В ситуации, когда точки на плоскости удалены друг от друга настолько, что $k$ совпадает с $n'$, вычислительная сложность *k*-средних составит $\sim O(k^2)$, что превышает таковую у выборки из БД. Таким образом, вычислительная сложность алгоритма № 2 составляет в худшем случае $\sim O(k^2)$.

### 2.2.4. Алгоритм № 3 кластеризационного метода для поиска взаимосвязей между кластерами – простыми воздействиями

Алгоритм № 3 предназначен для определения взаимосвязей между воздействиями на файлы и позволяет объединить несколько простых воздействий в одно сложное комплексное на основании значений поля «Имя файла» записей журнала $UsnJrnl с допущением о частичном совпадении имен и без зависимости от регистра символов. Алгоритм реализуется выполнением следующей последовательности действий:

1. По результатам работы алгоритмов № 1 и № 2 сформировать массив кластеров.

2. Последовательно обработать каждый кластер. На каждой итерации цикла:

   a. извлечь из элементов кластера (записей журнала) все имена файлов;
   b. произвести поиск совпадений имен файлов в остальных кластерах;
   c. отсортировать кластеры по возрастанию значения поля $T_f$ первой записи каждого кластера;
   d. последовательно объединить кластеры, если временной интервал между конечным элементом текущего кластера и начальным элементом следующего не превышает значения $t_1$;
   e. предложить объединение кластеров, если временной интервал между конечным элементом текущего кластера и начальным элементом следующего не превышает значения $t_2$;
   f. пометить все кластеры, которые участвовали в объединении на данной итерации цикла, как обработанные.

3. Вывести предложения об объединении кластеров, полученные на шаге 2 подпункте e алгоритма, специалисту, проводящему расследование инцидента ИБ.

Параметры $t_1$ и $t_2$ задаются в рамках предложенных значений специалистом, проводящим расследование инцидента ИБ, исходя из ожидаемой точности объединения кластеров записей журнала $UsnJrnl: чем больше значение $t_1$ и $t_2$, тем больше кластеров будет объединено. Диапазоны рекомендуемых значений параметров:

- $t_1$: 200 – 800 мс, оптимальное значение 400 мс;
- $t_2$: 2 – 8 с, значение по-умолчанию 3 с.

Графическое пояснение работы алгоритма № 3 представлено на рисунке 2.8.

Вычислительная сложность алгоритма определяется процедурой выборки записей журнала $UsnJrnl из БД и составляет $O(n\log_2 n)$, где $n$ – количество записей журнала.

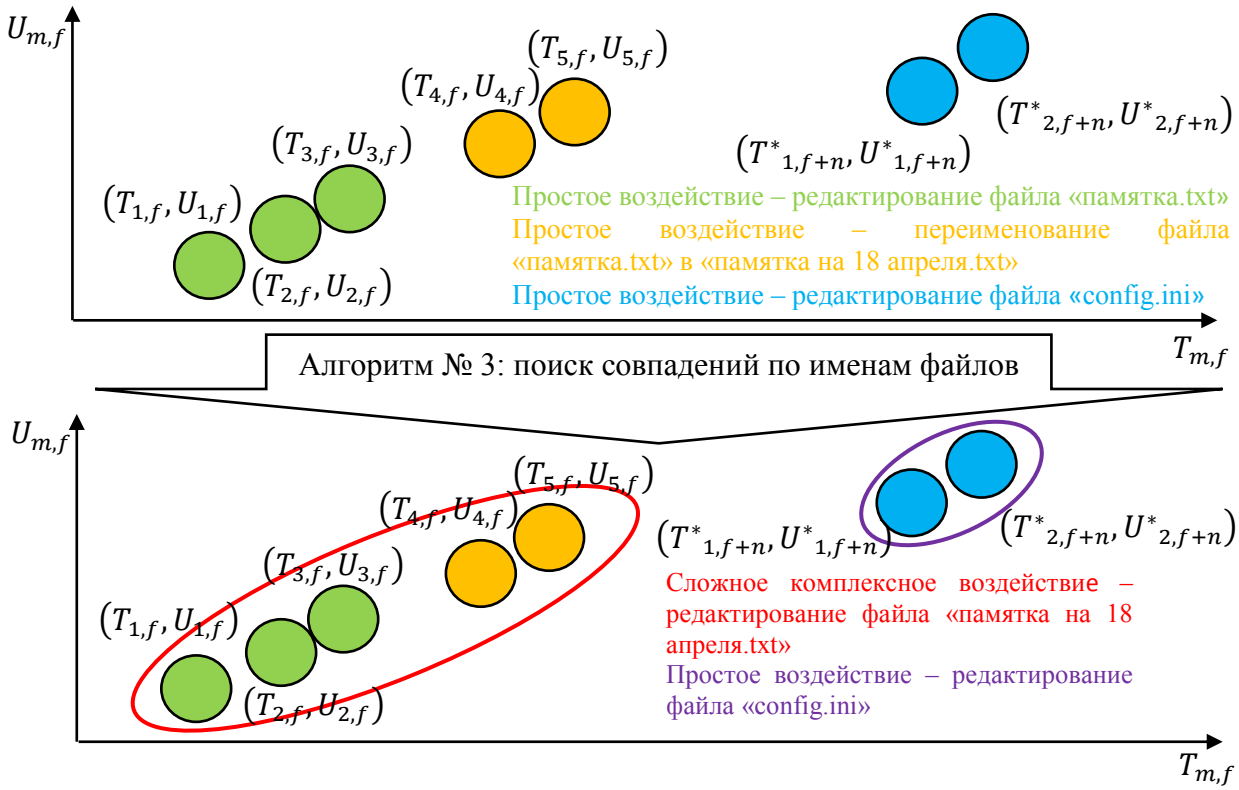


Рисунок 2.8. Графическое пояснение результата работы алгоритма № 3

### 2.2.5. Верификация данных журнала изменений тома $UsnJrnl

Верифицировать полученные воздействия на файлы возможно с применением событийной модели, описанный в пункте 2.1, на вход которой подаются:

- идентификаторы $I_{f1}$ и $I_{f2}$ начального и конечного состояний файла $f$;
- идентификаторы родительского каталога $D_{f1}$ и $D_{f2}$;
- имена файла $N_{f1}$ и $N_{f2}$;
- признак изменения содержимого $Z_f$.

Для функционирования сети Петри должна быть выставлена начальная маркировка $\mu$, соответствующая равенству заданных входных параметров и установке признака изменения содержимого. Параметр $Z_f$ можно определить исходя из значения поля $R_f$ записи, принадлежащей верифицируемому кластеру. Параметр $\tau$ вычисляется как разница между значениями полей $T_f$ смежных записей журнала $UsnJrnl, принадлежащих верифицируемому кластеру.

В результате проведения верификации специалист, проводящий расследование инцидента ИБ, проверяет последовательность записей в кластере на предмет наличия в них искажений. Стоит отметить, что записи в журнале $UsnJrnl формируются в соответствии с собственным алгоритмом драйвера файловой системы NTFS и не зависят от особенностей работы событийной модели. В связи с этим, может возникнуть несоответствие набора входных параметров искомому воздействию на файл в случае сложного комплексного воздействия. Специалист может убедиться в отсутствии искажений путем проверки последовательности значений полей $U_f$ – номера записей при переходе между файлами в рамках одного воздействия либо смежные, либо отличаются на 1-2 записи.

### 2.2.6. Порядок применения кластеризационного метода идентификации воздействий на файлы

Предлагаемый кластеризационный метод базируется на последовательном применении рассмотренных алгоритмов №№ 1 – 3. Для идентификации воздействий на файлы предложенным методом необходимо последовательно выполнить следующие действия:

1. Загрузить массив данных – журнал изменений тома $UsnJrnl, сохранив содержащиеся в нем записи в базу данных.

2. Создать множество выборок данных, разделив их по идентификаторам файлов $I_f$.

3. К каждой выборке данных применить алгоритм № 1, чтобы подготовить данные для проведения кластеризации алгоритмом *k*-средних.

4. Кластеризовать каждую подготовленную выборку алгоритмом *k*-средних, определив оптимальное количество кластеров $k$ в соответствии с алгоритмом № 2.

5. Сохранить кластеры, описывающие каждую выборку, в базу данных.

6. Применить алгоритм № 3 в целях поиска взаимосвязей между кластерами в базе данных для повышения точности идентификации сложных

комплексных воздействий. Применить/отклонить предложения по объединению кластеров, как результат работы алгоритма № 3.

7. Верифицировать кластеры с применением событийной модели.

8. Сформировать список идентифицированных воздействий на файлы, описанных кластерами записей журнала $UsnJrnl.

9. В сформированном списке найти те воздействия на файлы, которые имеют отношение к расследуемому инциденту ИБ.

Предложенный кластеризационный метод идентификации воздействий на файлы является универсальным и может быть применен к другим типам журналов при соблюдении следующих условий: наличие временной отметки появления записи в журнале, номера записи (допускается пронумеровать записи в порядке возрастания) и возможности разделения записей на блоки по дополнительному критерию (аналогично использованию $I_f$ в алгоритме №1 кластеризационного метода).

### 2.3. Метод экспресс-анализа событий, связанных с воздействиями на файлы

Кластеризованные записи журнала изменений тома $UsnJrnl могут быть использованы в качестве основы для сравнения записей журналов узлов инфраструктуры ИС при проведении экспресс-анализа воздействий на файлы в рамках расследования инцидента ИБ. Причем иметь все данные не требуется – достаточно проанализировать характерные значения в полях записей и подготовить шаблон[4] для каждого воздействия. В состав метода экспресс-анализа входит:

- алгоритм генерации шаблона воздействия на файл, применяемого до возникновения инцидента ИБ;
- алгоритм обнаружения и классификации воздействий с применением базы данных шаблонов;

[4] Под шаблоном воздействия на файл понимается декомпозиция данных, характеризующая это воздействие и позволяющая выделить его среди множества остальных воздействий.

- алгоритм обработки не совпавших с шаблонами записей журнала $UsnJrnl, которые применяются в процессе расследования.

### 2.3.1. Подготовка (генерация) шаблонов воздействий на файлы

Как уже упоминалось ранее, значения полей записей журнала являются основой шаблона *G*, который представляет собой совокупность фиксированных значений, описываемых вектором:

$$G = \{\langle G_{name}, G_{anomaly}, G_{priority}\rangle, \langle I_G, I_{sign}\rangle, \langle D_G, D_{sign}\rangle, \langle N_G, N_{sign}\rangle, \langle R_G, R_{sign}\rangle\}, \quad (13)$$

где в отношении к рассматриваемому шаблону воздействия:

- $G_{name}$ – название шаблона воздействия;
- $G_{anomaly}$ – признак аномальности воздействия;
- $G_{priority}$ – приоритет шаблона;
- $I_G$ – множество значений, описывающих идентификаторы файловых записей;
- $D_G$ – множество значений, описывающих идентификаторы родительских каталогов;
- $N_G$ – множество значений, описывающих имена файлов, расширения имен файлов и используемые специальные символы[5] (при их наличии);
- $R_G$ – множество значений, описывающих идентификаторы операций, полученных из поля «Идентификатор операции» (таблица 1.1) записей журнала;
- $I_{sign}$, $D_{sign}$, $N_{sign}$, $R_{sign}$ – признаки значимости соответствующих множеств значений $I_G$, $D_G$, $N_G$, $R_G$ для воздействия на файл, описываемого шаблоном *G*.

Говоря об аномальности воздействия, следует отметить, что этот параметр непосредственно связан с процедурой экспертной оценки и

[5] Символы, не являющиеся буквами, цифрами и пробелами, будем относить к категории специальных при рассмотрении имен файлов.

определяется специалистом-аналитиком, подготавливающим шаблон воздействия на файл с целью последующей их (воздействий) идентификации в рамках расследования инцидента ИБ. Например, создавая шаблоны воздействий на файлы, специалист, используя признак $G_{anomaly}$, может пометить шаблоны как аномальные, то есть не являющиеся санкционированными в процессе штатного функционирования ОС и активности пользователя.

Признаки значимости $I_{sign}$, $D_{sign}$, $R_{sign}$, $N_{sign}$ определяют необходимость использования значений из соответствующих множеств $I_G$, $D_G$, $R_G$, $N_G$ при экспресс-анализе воздействий на файлы с использованием шаблонов. Каждый признак значимости может принимать значение 0 или 1, определяемое специалистом-аналитиком, подготавливающим шаблон, в зависимости от того, важны ли значения соответствующего множества при идентификации воздействия на файл с применением генерируемого шаблона. Стоит отметить, что поля, не выделенные как значимые, не будут учитываться в последующем применении шаблона воздействия для его идентификации в рамках экспресс-анализа событий. В свою очередь, чем точнее указаны значения полей шаблона воздействия, тем ниже вероятность появления ошибок 1 и 2 рода при проведении экспресс-анализа событий ИБ с применением шаблонов.

На основании значения приоритета $G_{priority}$ выстраивается очередь шаблонов идентификации воздействий на файлы – чем выше числовое значение приоритета, тем раньше в очереди располагается шаблон.

Искомые множества $I_G$, $D_G$ и $R_G$ формируются автоматически посредством выборки уникальных числовых значений соответствующих полей после применения кластеризационного метода идентификации воздействий на файлы.

Ввиду того, что значения из множества $N_G$ представляют собой не числа, а наборы символов, то в целях повышения точности идентификации

воздействий на файлы с применением шаблонов представим $N_G$ как совокупность подмножеств $N_{ed}$, $N_{ext}$, $N_{ft}$ и $N_{sym}$, где:

- $N_{ed}$ – подмножество значений, характеризующих «степень различия» между именами файлов;
- $N_{ext}$ – подмножество значений, описывающих расширения имен файлов;
- $N_{ft}$ – подмножество значений, описывающих имена файлов;
- $N_{sym}$ – подмножество значений, описывающих специальные символы, содержащиеся в именах файлов.

Значения, принадлежащие каждому из подмножеств $N_{ed}$, $N_{ext}$, $N_{ft}$ и $N_{sym}$, могут быть значимы для шаблона независимо друг от друга. В связи с этим, значение $N_{sign}$ представляется как совокупность значений признаков $N_{ed_{sign}}$, $N_{ext_{sign}}$, $N_{ft_{sign}}$, $N_{sym_{sign}}$.

Множество $N_{ft}$ содержит уникальные значения имен файлов, полученных из полей записей, составляющих идентифицированное воздействие.

Значения из множества $N_{ext}$ получаются путем обработки содержимого поля $N_{ft}$ посредством выделения части символов, стоящих за последним символом точки. Например, расширением имени «config.ini» является «ini», а для «config.ini.bak» – «bak».

Также следует выделить значения $N_{sym}$, из значений, принадлежащих $N_{ft}$, осуществив проверку каждого символа на принадлежность к категории специальных.

Для определения «степени различия» между именами в целях формирования значений множества $N_{ed}$ необходимо получить количественную величину, которая характеризует, насколько два имени

файла в рамках одного воздействия отличаются друг от друга. Так как имена файлов представляют собой набор символов в виде строки, то «степень различия» между именами интерпретируется как расстояние между строками *d*. В работах [138, 139] рассмотрен подробный анализ алгоритмов сравнения строк и приведены примеры метрик (перечень примеров не является исчерпывающим), используемых для определения *d*:

- метрика Левенштейна: $0 \le d(N_{ft_1}, N_{ft_2}) \le \max(\left|N_{ft_1}\right|, \left|N_{ft_2}\right|)$, где $N_{ft_1}$ – первое имя файла в рамках идентифицированного воздействия, $N_{ft_2}$ – сравниваемое имя. При расчете расстояния допускаются символьные операции замены, удаления, добавления;
- метрика Хэмминга: $0 \le d(N_{ft_1}, N_{ft_2}) \le \left|N_{ft_1}\right|$. При расчете расстояния допускаются символьные операции замены;
- «эпизодическая» метрика: $d(N_{ft_1}, N_{ft_2}) = \begin{cases} \left|N_{ft_2}\right| - \left|N_{ft_1}\right| \\ \infty \end{cases}$. При расчете расстояния допускаются символьные операции вставки;
- метрика *q*-грамм: $d(N_{ft_1}, N_{ft_2}) = \sum_{\omega \in \Omega^q} \left| \nu_{ft_1}[\omega] - \nu_{ft2}[\omega] \right|$ [138], где: $\Omega$ – алфавит, включающий в себя все символы, допустимые при именовании файлов в файловой системе NTFS; $\Omega^q$ – множество всех *q*-грамм, состоящих из символов алфавита $\Omega$; $\omega$ – *q*-грамма, $\omega \in \Omega$; $\nu_{ft_1}[\omega]$ и $\nu_{ft2}[\omega]$ – количество вхождений *q*-грамм $\omega$ в $N_{ft_1}$ и $N_{ft2}$ соответственно. При расчете расстояния будем применять биграммы ($q$=2).

Для решения задачи расчета расстояния между строками, т.е. определения, насколько одно имя отличается от другого, необходимо выбрать тип метрики. При выборе следует учитывать, что имена могут отличаться полностью. Примеры расчета расстояний между первым именем в идентифицированном воздействии и всеми остальными указаны в таблице 2.5.

Таблица 2.5. Примеры расчета расстояний между именами для различных идентифицированных воздействий

| **Идентифицированные воздействия** | **Имена файла(-ов) в рамках воздействия / исходное имя** | **Расстояние**[6] | | | |
|---|---|---|---|---|---|
| | | **Л** | **Х** | **Э** | ***q*** |
| Редактирование «офисного» документа в текстовом процессоре Microsoft Word | ~$кладная.docx / Накладная.docx | 2 | 2 | ∞ | 4 |
| | WRD0000.tmp / Накладная.docx | 14 | – | ∞ | 23 |
| | WRL0001.tmp / Накладная.docx | 14 | – | ∞ | 23 |
| Шифрование файла вредоносным программным обеспечением Jigsaw | Photo0001.bmp.fun / Photo0001.bmp | 4 | – | ∞ | 4 |
| Шифрование файла вредоносным программным обеспечением WannaCry | Photo01.bmp.WNCRYT / Photo01.bmp | 7 | – | ∞ | 7 |
| | Photo01.bmp.WNCRY / Photo01.bmp | 6 | – | ∞ | 6 |
| Изменение расширения загруженного текстового файла с целью внедрения и запуска php кода | research.txt / research.php | 3 | 3 | ∞ | 6 |

Из таблицы 2.5 видно, что в рамках одного идентифицированного воздействия на файл возможно как полное несовпадение имен, так и различия в 1-2 символах. Выбор типа метрики влияет на скорость работы алгоритма расчета расстояния и затраты памяти для хранения результатов вычислений. В процессе генерации шаблонов воздействий будем руководствоваться принципом учета минимального значения расстояния *d* между строками: если расстояние в нескольких одинаковых воздействиях на файл отличается (непостоянно) и/или равно длине имени (полное несовпадение имен), то его не следует учитывать в шаблоне. Наоборот, если расстояние в нескольких одинаковых воздействиях на файл не отличается (постоянно), то оно должно быть использовано при генерации шаблона воздействия на файл. С позиции минимального значения расстояния *d,* скорости работы алгоритма расчета расстояния между строками и затрат

[6] Рассмотрены вышеуказанные метрики: Л – Левенштейна, Х – Хэмминга, Э – «эпизодическая», *q* – *q*-граммы (при *q*=2).

памяти для хранения результатов вычислений оптимальным выглядит выбор метрики Левенштейна.

Блок-схема алгоритма генерации шаблона воздействия на файл и записи его в базу данных шаблонов представлена на рисунке 2.9.

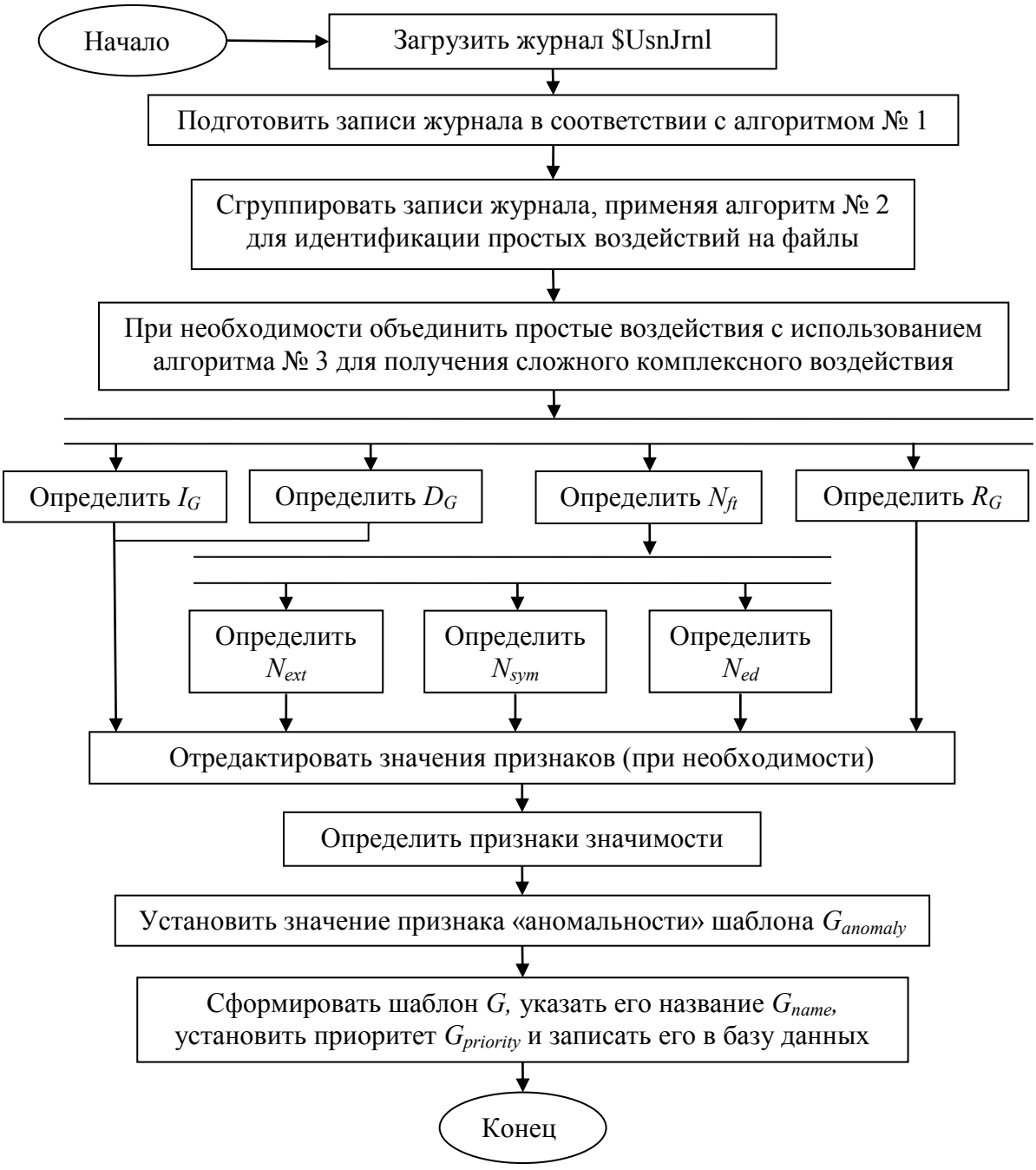


Рисунок 2.9. Блок-схема алгоритма генерации шаблона воздействия на файл

Вычислительная сложность алгоритма генерации шаблонов определяется процедурой выборки записей журнала $UsnJrnl из БД и составляет $O(n'\log_2 n')$, где $n'$ – количество записей журнала в блоке.

В соответствии с предложенным на рисунке 2.9 алгоритмом разберем пример создания шаблона ранее рассмотренного сложного комплексного идентифицированного воздействия. На рисунке 2.10 показан пример генерации шаблона, соответствующего идентифицированному воздействию на исполняемый файл – создание, изменение содержимого и запуск исполняемого файла WannaCry.

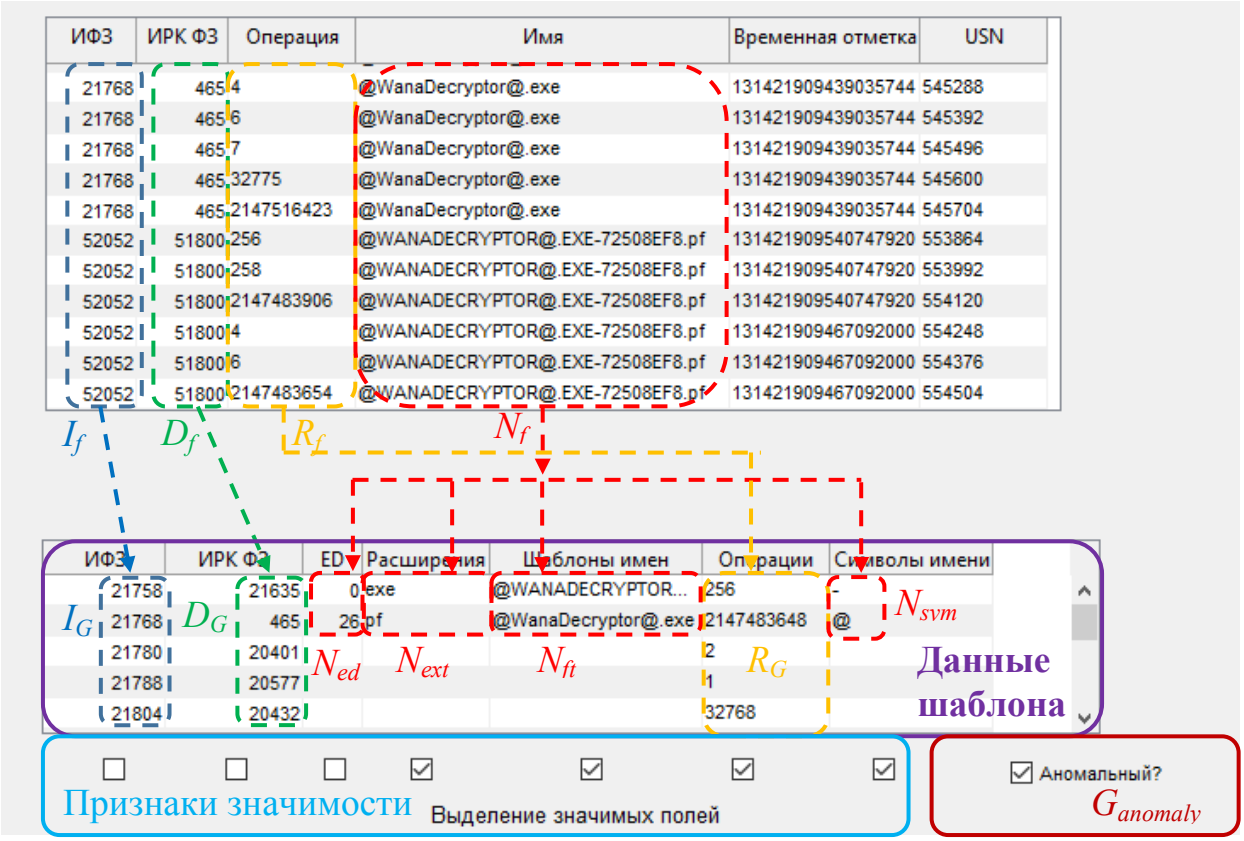


Рисунок 2.10. Пример генерации шаблона воздействия – создание, изменение содержимого и запуск исполняемого файла WannaCry

На рисунке 2.10 продемонстрированы соответствия компонентов вектора $G$ шаблона полям записей журнала $UsnJrnl, представленных в табличном виде. Поля «Временная отметка» и USN соответствуют $T_f = \{T_{m,f} \mid m = \overline{1,M}\}$ и $U_f = \{U_{m,f} \mid m = \overline{1,M}\}$. Признаки значимости $I_{sign}$, $D_{sign}$, $R_{sign}$, $N_{ed_{sign}}$, $N_{ext_{sign}}$, $N_{ft_{sign}}$, $N_{sym_{sign}}$ соответствуют тем данным, под которыми расположен маркер √.

Каждый сгенерированный шаблон сохраняется в базу данных для последующего использования алгоритмом обнаружения и классификации воздействий.

### 2.3.2. Алгоритм обнаружения и классификации воздействий на файлы с применением базы данных шаблонов

Обнаружение и классификация воздействий на файлы с использованием шаблонов осуществляется на основе нахождения совпадений в записях журнала изменений тома \$UsnJrnl с данными шаблонов. Как отмечалось выше, воздействие на файл описывается шаблоном, т.е. значениями из множеств $I_G$, $D_G$, $R_G$, $N_G$, которые можно рассматривать как компоненты вектора $G_f$, характеризующего воздействие на файл. Между векторами $V_f$ и $G_f$ существует взаимосвязь ($G_f = g(V_{if}), i = \overline{1, M}$), которая основана на рассмотренных кластеризационном методе идентификации воздействий на файлы и алгоритме генерации шаблонов воздействий.

Поскольку в контексте определенного воздействия те или иные компоненты могут быть значимыми или незначимыми и иметь различные значения из соответствующих множеств, то сравнение с шаблоном осуществляется пороговым способом по степени близости, рассчитываемой как взвешенное скалярное произведение компонент вектора $G_f$ с соответствующими компонентами шаблона $G$.

Алгоритм обнаружения и классификации воздействий на файлы реализуется выполнением следующей последовательности действий:

1. Открыть базу данных шаблонов. Загрузить совокупность хранящихся в базе шаблонов, отсортировать шаблоны по убыванию приоритетов $G_{priority}$ и начать цикл по поиску соответствия шаблонов в наборе записей журнала.

2. Загрузить из выбранного в цикле шаблона компоненты вектора *G*.

3. Сформировать вектор-столбец максимальных весовых коэффициентов $W_{\max}$ (все компоненты вектора имеют ненулевое значение и по умолчанию имеют вес, равный 1). $W_{\max}$ описывает ситуацию, когда все компоненты вектора $G$, характеризующие воздействие, являются значимыми.

4. На основе признаков значимости $I_{sign}$, $D_{sign}$, $R_{sign}$, $N_{sign}$ сформировать вектор-строку весовых коэффициентов $W$ (по умолчанию вес равен 1). $W$ описывает ситуацию, когда значимость компонентов вектора $G$, характеризующих файл, определяется соответствующими признаками.

5. Умножением вектор-строки $W$ на вектор-столбец $W_{\max}$ рассчитать пороговое значение $G_{threshold}$.

6. Выбрать из журнала записи в соответствии со значимыми признаками шаблона $G$.

7. Разделить выбранные записи на блоки по временным интервалам $t$ для выделения воздействий из общего набора записей.

8. Сравнить полученные блоки записей с текущим шаблоном $G$: для каждого признака текущего шаблона $G$ найти вхождения (совпадения) значений признака в полях записей из блоков и сформировать вектор-столбец вхождений $W_{similar}$.

9. Рассчитать значение коэффициента сходства $G_{similar}$, умножив вектор-строку $W$ на вектор-столбец $W_{similar}$.

10. Сравнить $G_{similar}$ и $G_{threshold}$: если $G_{similar} \geq G_{threshold}$, то набор записей в блоке совпадает с шаблоном $G$.

11. Принять решение об условной аномальности воздействия: если записи в блоке принадлежат временному интервалу инцидента ИБ, то классифицировать их как условно аномальные.

12. На основании признака аномальности $G_{anomaly}$ принять решение о классификации воздействия, совпавшего с шаблоном $G$, как аномального.

13. Перейти к следующему шаблону.

Блок-схема алгоритма представлена на рисунке 2.11.

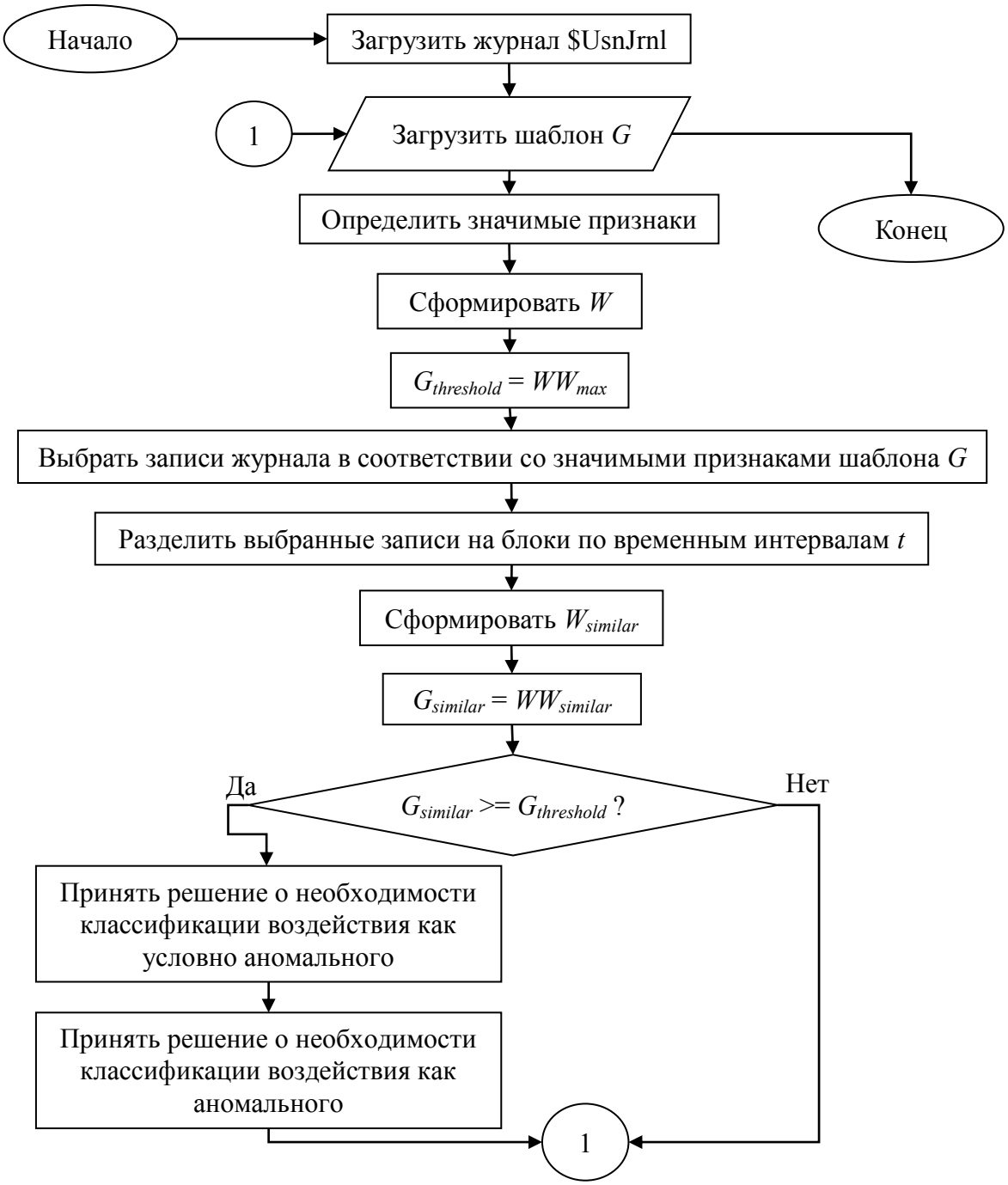


Рисунок 2.11. Блок-схема алгоритма обнаружения и классификации воздействий на файлы с применением шаблонов

Говоря об оценке вычислительной сложности предложенного алгоритма следует учитывать, что наибольший вклад вносит процедура выборки записей журнала в соответствии с значимыми признаками шаблона *G,* вычислительная сложность которой, за счёт использования двоичных деревьев поиска в индексах базы данных, хранящей записи загруженного

журнала, составляет $O(n\log_2 n)$, где $n$ – количество записей журнала. В сравнении с другими популярными алгоритмами классификации, например, SVM ($O(n^2)$), или деревьям решений ($O(h+n\log_2 n)$, где $h$ – «глубина» дерева), предложенный алгоритм показывает лучшие или сопоставимые результаты, но устойчив к повышению вычислительной сложности из-за увеличения размерности пространства признаков (компонентов вектора $G$), а также позволяет менять дополнительные условия аномальности и значимые признаки без необходимости переобучения классификатора.

### 2.3.3. Алгоритм обработки записей журнала, не совпавших с шаблонами

Не совпавшие с шаблонами записи журнала обрабатываются в соответствии алгоритмом, который реализуется выполнением следующей последовательности действий:

1. Выбрать не совпавшие с шаблонами записи журнала.

2. Разделить записи на блоки по временным интервалам $t$ для выделения воздействий из общего набора записей. Длительность интервала не должна превышать 500 – 2000 мс.

3. На основании значений поля $R_f$ записей в блоке указать действия, совершенные по отношению к файлам, в соответствии с [75]:

- при $R_f = 256$ — создание файла;
- при $R_f = 512$ — удаление файла;
- при $R_f = 4096$ и $R_f = 8192$ — переименование файла;
- при $R_f$ = 1, 2, 4, 16, 32, 64, 2097152, 4194304 — изменение содержимого файла;
- при $R_f$ = 1024, 2048, 16384, 32768, 65536, 1048576 — изменение служебной информации о файле.

Указанный алгоритм предназначен для группировки записей журнала, которые не совпали с шаблонами в БД, с целью сокращения объема анализируемой информации.

### 2.3.4. Структура метода экспресс-анализа событий ИБ, связанных с воздействиями на файлы

КС работают под управлением многозадачных систем, вследствие чего, специалист-аналитик вынужден исследовать объем информации, измеряющийся сотнями тысяч записей, о воздействиях на файлы, связанных как с процессом штатной обработки информации (санкционированные), так и с действиями злоумышленника (несанкционированные), зачастую «сплетенных» в едином массиве данных. Последовательное выполнение этапов метода экспресс-анализа позволяет определить события и сконцентрировать внимание специалиста, проводящего расследование инцидента ИБ, только на тех из них, которые связаны с несанкционированными воздействиями на файлы. Поэтапная схема метода экспресс-анализа событий ИБ представлена на рисунке 2.12.

Для сокращения объема анализируемой информации необходимо разделить (классифицировать) множества осуществленных воздействий на нормальные, условно аномальные и аномальные. Нормальные воздействия не имеют отношения к инциденту ИБ и должны быть исключены из отчета специалистом-аналитиком, проводящим расследование. Условно аномальные воздействия должны быть рассмотрены в рамках инцидента в том случае, если удовлетворяют дополнительным критериям, например, дата и время начала/окончания, расположение файлов. Аномальные воздействия не должны возникать в процессе штатного функционирования КС и должны быть проанализированы в приоритетном порядке.

Предложенный метод экспресс-анализа событий ИБ универсален и может быть использован в других типах ОС и ФС с учетом особенностей изменения признаков, характеризующих файл, которые должны быть отражены в шаблоне воздействий.

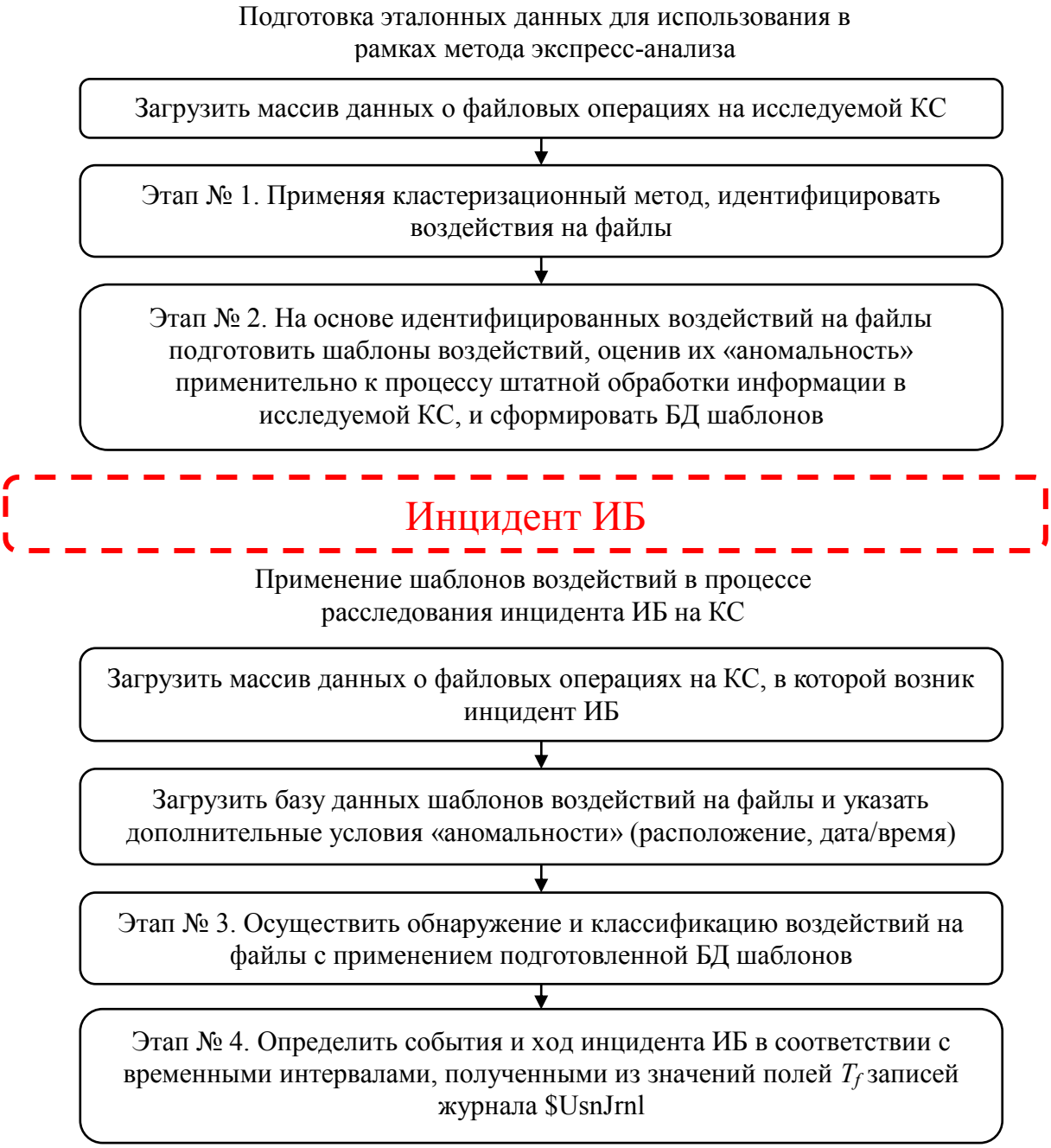


Рисунок 2.12. Поэтапная схема метода экспресс-анализа событий ИБ

Рассмотрим программный комплекс, который реализует алгоритмы из представленных методов.

### 2.4. Выводы по главе 2

1. Обоснована возможность описания процесса идентификации воздействий на файлы, в том числе направленных на нарушение действующей политики безопасности, с использованием математического

аппарата сетей Петри на основе формализованного набора признаков, характеризующих файл.

2. Предложенный кластеризационный метод позволяет автоматизировать процесс идентификации воздействий на файлы посредством применения адаптированных алгоритмов подготовки входных данных и определения оптимального количества кластеров. Отличием предложенного метода от существующих является автоматический подбор и оценка оптимальных параметров кластеризации.

3. Представленный метод экспресс-анализа позволяет автоматизировать процесс выявления нормальных, условно аномальных и аномальных воздействий на файлы, лежащих в основе событий ИБ. Отличием предложенного метода от существующих является использование базы данных шаблонов воздействий на файлы, позволяющих добавлять / изменять классы воздействий без необходимости проведения ресурсоемкой процедуры переобучения классификатора.

# Глава 3. Разработка комплекса программных средств идентификации воздействий на файлы при расследовании инцидентов информационной безопасности

Представленные во второй главе алгоритмы и методы легли в основу комплекса программных средств, который предназначен для автоматизации процесса идентификации воздействий на файлы при расследовании инцидентов ИБ.

В состав комплекса входят 3 программных средства:

- генерации шаблонов воздействий на файлы, используемое при расследовании инцидентов информационной безопасности;
- обнаружения событий информационной безопасности, использующее шаблоны воздействий на файлы;
- генерации компьютерных атак и осуществления воздействий на файлы.

Рассмотрим назначение каждого программного средства, выбранную платформу для разработки, порядок работы с ними и процесс взаимодействия между программными средствами.

## 3.1. Программное средство генерации шаблонов воздействий на файлы, используемое при расследовании инцидентов ИБ

Программное средство предназначено для реализации кластеризационного метода идентификации воздействий на файлы. Платформой для разработки является программная среда MATLAB. Структурно программное средство состоит из трех частей:

- ПО для извлечения журнала изменений тома $UsnJrnl;
- ПО обработки записей журнала изменений тома $UsnJrnl;
- ПО генерации шаблонов воздействий на файлы.

### 3.1.1. Программное обеспечение для извлечения журнала изменений тома $UsnJrnl и программное обеспечение обработки записей журнала

Задачей первого ПО является получение копии потока данных $J файла журнала изменений тома $UsnJrnl для последующей его обработки. В связи с тем, что файл является системным, драйвер ФС запрещает прямой доступ к нему. При работе с МНИ при неактивной ОС копирование потока данных $J файла журнала изменений тома $UsnJrnl возможно осуществить с использованием шестнадцатеричного редактора в обход драйвера ФС. На работающей ОС необходимо использовать специальное ПО, в данной работе применяется ExtractUsnJrnl [140], которое позволяет извлекать поток данных $J как с текущего МНИ, так и из образа. В связи с тем, что $UsnJrnl является разреженным файлом, то при копировании потока данных $J шестнадцатеричным редактором часть содержимого, которая не содержит записей, окажется заполнена нулями. Тем не менее, данная особенность не влияет на результат последующей обработки записей.

Второе ПО преобразует двоичный массив данных журнала изменений тома $UsnJrnl в БД формата SQLite [141], позволяя работать с записями журнала с использованием языка SQL.

### 3.1.2. Программное обеспечение генерации шаблонов воздействий на файлы

Рассматриваемое ПО разработано с использованием среды MATLAB. Задачей ПО является загрузка и анализ записей журнала изменений тома $UsnJrnl, хранящихся в БД SQLite, полученного с МНИ стендового компьютера. ПО реализует представленный в п. 2.2 кластеризационный метод идентификации воздействий на файлы, осуществляет верификацию полученных воздействий, согласно событийной модели (п. 2.1), и выполняет генерацию шаблонов воздействий на файлы.

Главное окно ПО представлено на рисунке 3.1. Область №1 предназначена для отображения меню ПО. Меню «Файл» позволяет выбрать

БД с записями журнала и запустить кластеризационный метод идентификации воздействий на файлы, а также открыть существующую БД шаблонов воздействий на файлы и запустить ПО извлечения журнала изменений тома $UsnJrnl и ПО обработки записей журнала. Интерфейс с параметрами запуска ПО извлечения журнала и обработки записей показан на рисунке 3.2.

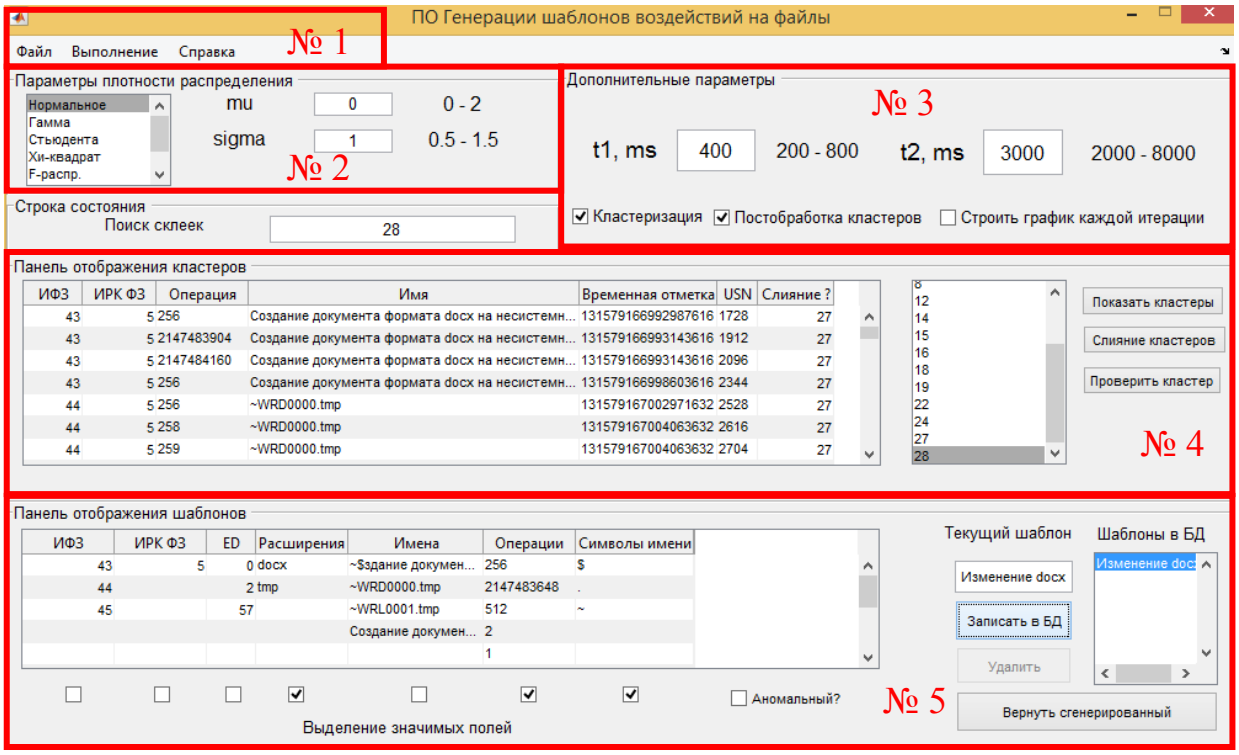


Рисунок 3.1. Интерфейс ПО генерации шаблонов воздействий

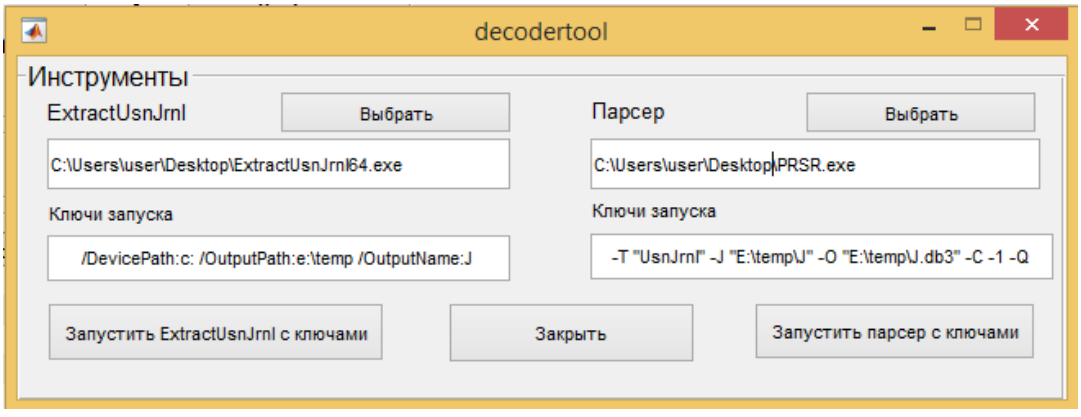


Рисунок 3.2. Интерфейс запуска ПО извлечения журнала изменений тома $UsnJrnl и обработки записей журнала

Меню «Выполнение» позволяет повторно запустить кластеризационный метод, если результаты предыдущей кластеризации не удовлетворяют специалиста, который готовит шаблоны воздействий на

файлы. В этом же меню находится пункт для запуска процедуры генерации шаблона (с последующим отображением шаблона в области № 5) на основе выбранного кластера и процедуры очистки БД шаблонов.

Область задания параметров PDF обозначена как № 2 на рисунке 3.1. В списке рассматриваемой области присутствуют те разновидности PDF, которые указаны в таблице 2.4. Для каждой PDF предусмотрены графические элементы для задания параметров распределений и диапазон их оптимальных значений.

Область № 3 дополнительных параметров позволяет указать значения параметров $t_1$ и $t_2$, необходимых для работы алгоритма № 3 кластеризационного метода. В этой области также можно указать режим работы ПО: пункт «Кластеризация» включает/отключает применение алгоритмов № 1 и 2 кластеризационного метода (если в рамках генерации шаблонов возникает необходимость исключить кластерный анализ), а пункт «Постобработка кластеров» соответственно включает/отключает алгоритм № 3. Пункт «Строить график каждой итерации» позволяет увидеть расчет оптимального количества кластеров $k$, а также расположение центроидов относительно скопления точек для каждого значения идентификатора файловой записи $I_f$. Пример расположения центроидов и скоплений точек и показан на рисунке 3.3.

Область № 4 предназначена для отображения воздействий на файлы – кластеров, состоящих из точек на плоскости, соответствующих записям журнала изменений тома \$UsnJrnl. На рисунке 3.1 столбцы таблицы соответствуют следующим полям записей журнала \$UsnJrnl:

- столбец «ИФЗ» содержит идентификаторы файла $I_f$;
- столбец «ИРК ФЗ» содержит идентификатор родительского каталога $D_f$;
- столбец «Операция» содержит идентификаторы действий $R_f$, совершенных по отношению к файлам;
- столбец «Имя» содержит имена файлов $N_f$;

- столбец «Временная отметка» содержит время появления записи в журнале $T_f$ в формате FILETIME в десятичной системе счисления;
- столбец «USN» содержит номера записей $U_f$;
- столбец «Слияние ?» содержит номер кластера, с которым следует объединить рассматриваемый кластер при положительном решении специалиста, готовящего шаблон воздействия на файл.


Рисунок 3.3. Расчет оптимального количества кластеров *k* и построение центроидов и скоплений точек

Кнопка «Показать кластеры» отображает расположение центроидов и скоплений точек на плоскости для каждого значения $I_f$, аналогично рисунку 3.3. Кнопка «Слияние кластеров» открывает форму для объединения кластеров, представленную на рисунке 3.4. В элементе «Панель объединения» указан список «Предложение» с перечнем объединяемых кластеров, а также список «Объединить», который наполняет специалист путем выбора элемента из списка «Предложение» и нажатия кнопки

«Объединить». Процедура объединения кластеров осуществляется по нажатию кнопки «Применить изменения к БД».

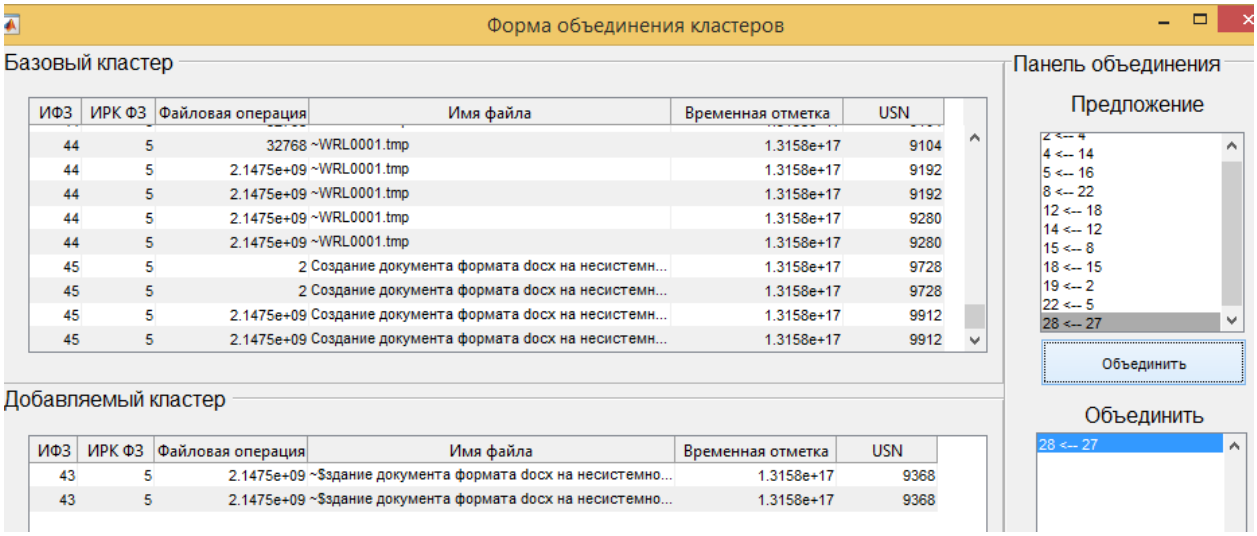


Рисунок 3.4. Форма объединения кластеров

Кнопка «Проверить кластер» запускает процедуру верификации полученной совокупности записей в соответствии с событийной моделью идентификации воздействий на файлы. Результат проверки отображается в виде всплывающего сообщения, как показано на рисунке 3.5.

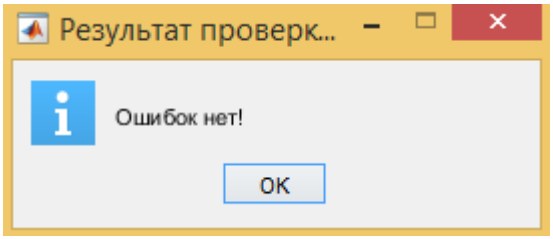


Рисунок 3.5. Результат проверки совокупности записей журнала изменений тома $UsnJrnl согласно событийной модели

Область № 5 предназначена для отображения значений компонентов сгенерированного шаблона (в соответствии с выражением (13)), возможности установки значимых признаков шаблона, установки признака «аномальности» и имени шаблона. В правой части области указан список сохраненных в БД шаблонов.

IDEF0-схема [142], описывающая взаимодействие модулей программного средства, представлена на рисунке 3.6.

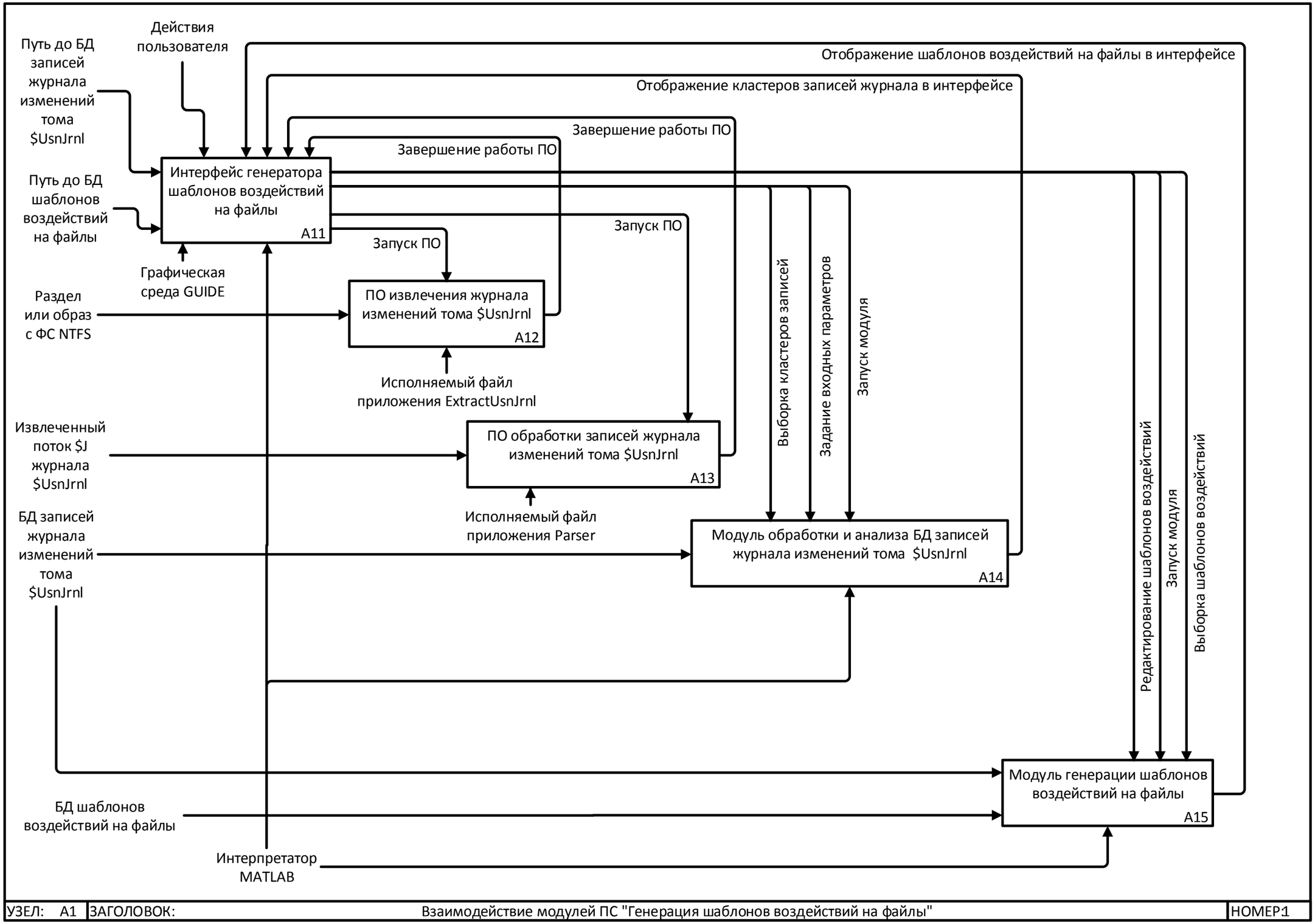


Рисунок 3.6. IDEF0-схема программного средства генерации шаблонов воздействий на файлы

### 3.2. Программное средство обнаружения событий ИБ, использующее шаблоны воздействий на файлы

Программное средство предназначено для реализации метода экспресс-анализа событий ИБ, связанных с воздействиями на файлы. Платформой для разработки является программная среда Python. Структурно программное средство состоит из трех частей:

- ПО для извлечения журнала изменений тома $UsnJrnl (п. 3.1.1);
- ПО обработки записей журнала изменений тома $UsnJrnl (п. 3.1.1);
- ПО обнаружения и классификации воздействий на файлы, лежащих в основе событий ИБ.

#### 3.2.1. Программное обеспечение обнаружения и классификации воздействий на файлы, лежащих в основе событий ИБ

Рассматриваемое ПО разработано с использованием среды Python. Задачей ПО является анализ записей журнала изменений тома $UsnJrnl, полученных с исследуемого МНИ. ПО реализует представленный в п. 2.3 алгоритм обнаружения и классификации воздействий на файлы из состава метода экспресс-анализа. Главное окно ПО представлено на рисунке 3.7.

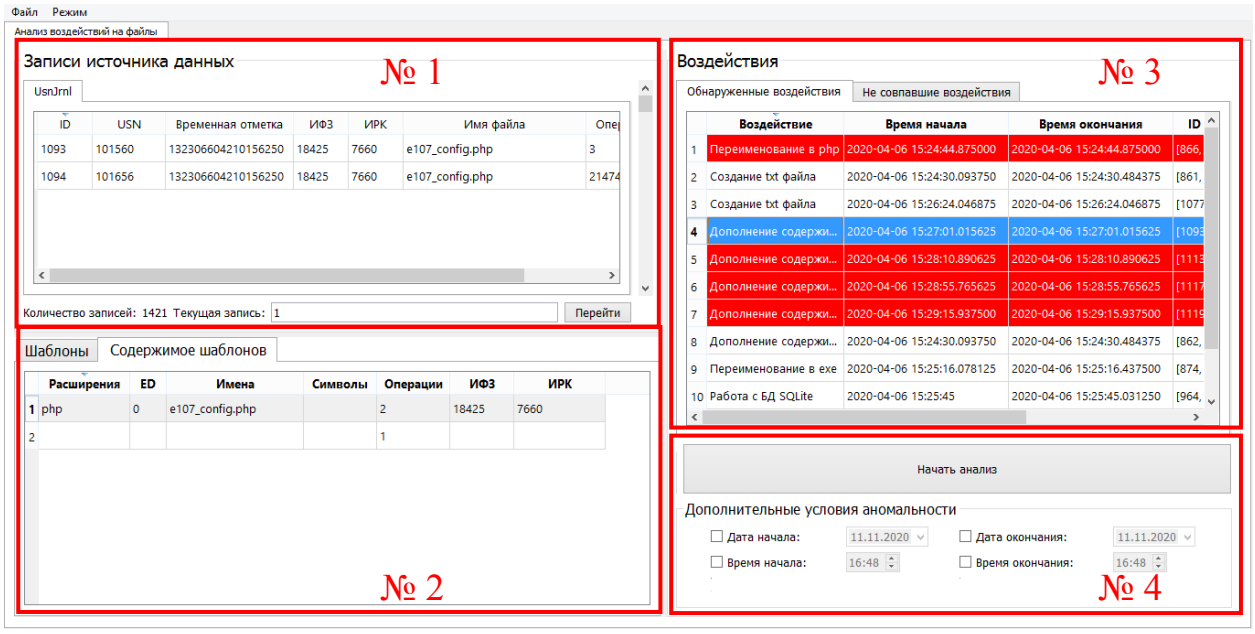


Рисунок 3.7. Интерфейс ПО обнаружения и классификации воздействий на файлы, лежащих в основе событий ИБ

Область, предназначенная для отображения записей журнала изменений тома $UsnJrnl обозначена как № 1 на рисунке 3.7. По-умолчанию в таблице отражены все записи $UsnJrnl, полученные из БД SQLite с исследуемого МНИ.

В области № 2 представлены список шаблонов и данные каждого шаблона, полученные из БД шаблонов воздействий на файлы. Область необходима для того, чтобы специалист мог определить корректность обнаружения и классификации воздействия в соответствии с шаблоном.

При выборе идентифицированного воздействия из таблицы в области № 3 в таблице области № 1 отображаются записи журнала, лежащие в основе этого воздействия, а в списке шаблонов соответственно выбирается тот, на основе которого произведены обнаружение и классификация воздействия. Красным цветом в таблице отображаются аномальные воздействия, белым — нормальные, желтым — условно аномальные, т.е. те, которые удовлетворяют дополнительным условиям аномальности, а именно временным интервалам начала и окончания воздействия.

Те записи журнала $UsnJrnl, содержимое которых не совпало с шаблонами, обрабатываются с целью идентификации воздействий на файлы в соответствии с алгоритмом, представленным в п. 2.3.3. У каждого воздействия указываются тип совершенных файловых операций, что позволяет специалисту оценить последствия воздействия. Пример таких воздействий представлен на рисунке 3.8.

Область № 4 предназначена для запуска ПО и задания дополнительных условий аномальности в виде ограничения временных интервалов, когда предположительно осуществлялись воздействия на файлы, лежащие в основе инцидента ИБ.

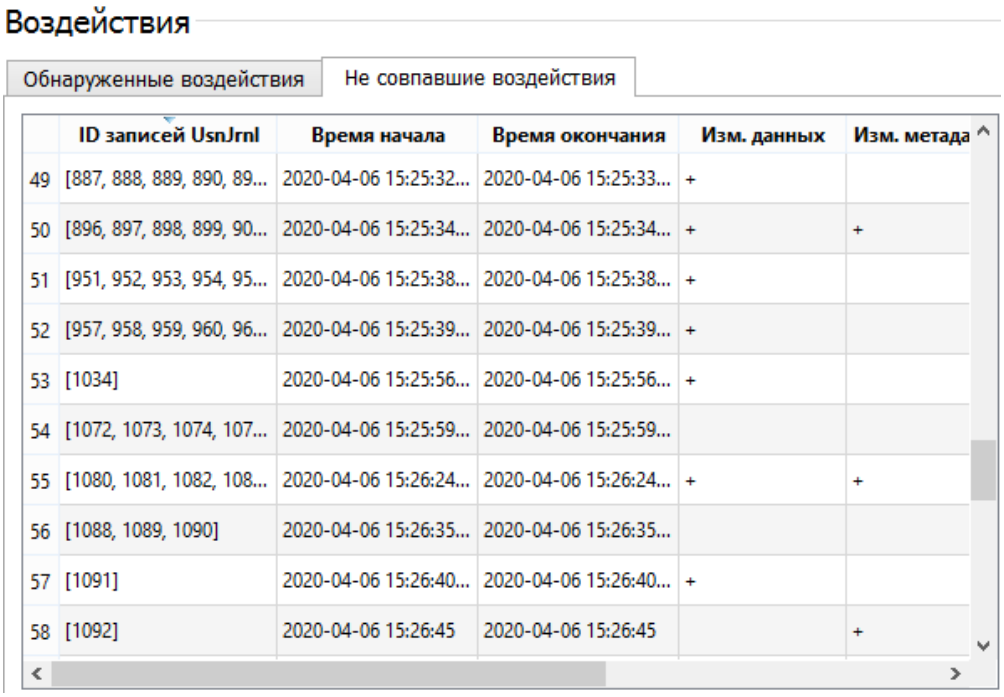


Рисунок 3.8. Таблица с воздействиями на файлы, записи которых не совпали с шаблонами

По результатам обнаружения и классификации воздействий в рассматриваемом ПО предусмотрено меню с рекомендациями по совершенствованию мер, направленных на обеспечение ИБ. Набор рекомендаций привязан к шаблону и составляется в момент создания шаблона. Пример меню рекомендаций приведен на рисунке 3.9.

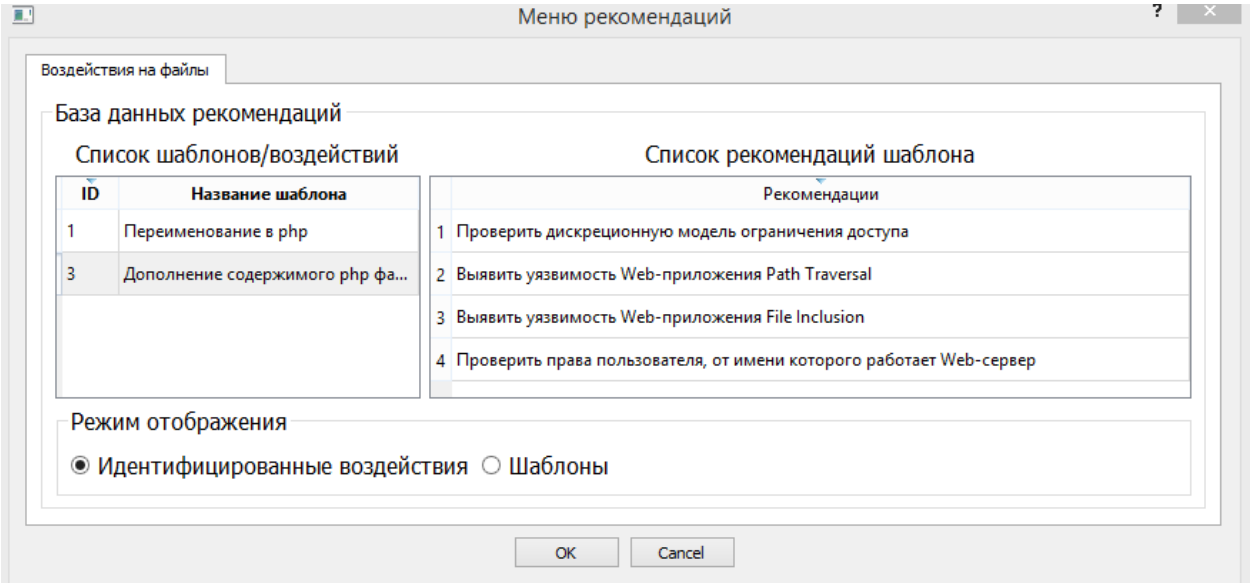


Рисунок 3.9. Пример меню рекомендаций

IDEF0-схема, описывающая взаимодействие модулей программного средства, представлена на рисунке 3.10.

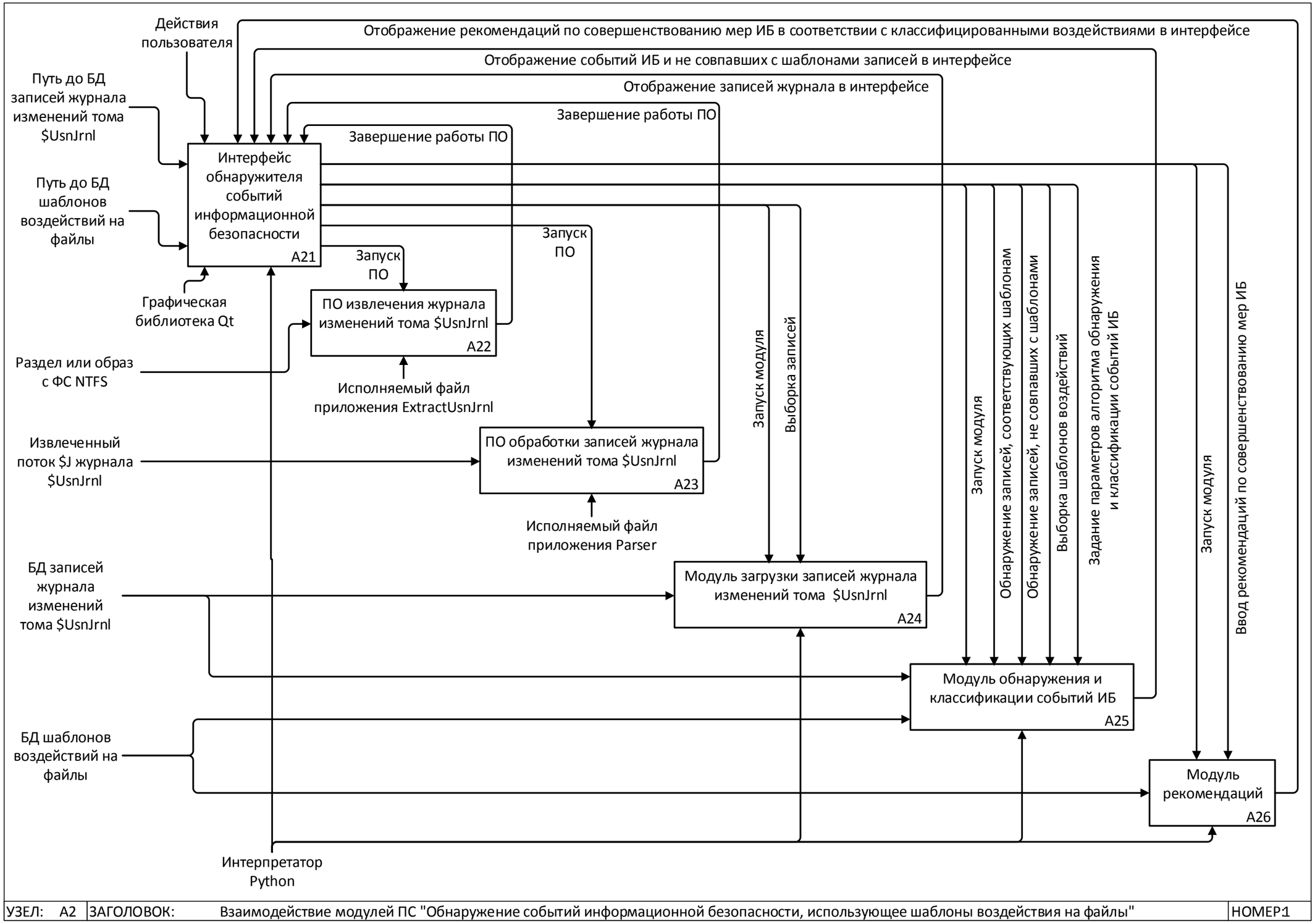


Рисунок 3.10. IDEF0-схема программного средства обнаружения и классификации событий ИБ

### 3.3. Программное средство генерации компьютерных атак и осуществления воздействий на файлы

Программное средство предназначено для проведения КА на ОС стендового компьютера в целях имитации событий ИБ, лежащих в основе инцидента. В качестве основы предлагается использовать инструмент Metasploit, входящий в состав дистрибутивов для проведения тестов на проникновение, таких как Kali Linux, Parrot, Black Arch.

При решении задачи генерации шаблонов воздействий на файлы специалисту необходимо загрузить в Metasploit скрипты, которые приведут к воздействиям на файлы. Впоследствии, например при изменении конфигурации сервисов для расширения набора воздействий, не возникает необходимости в повторной имитации всех КА – достаточно лишь дополнить набор скриптов. Использование программного средства генерации КА позволяет автоматизировать процесс осуществления воздействий на файлы, имитируя реальную активность злоумышленника или ВПО.

### 3.4. Методика использования комплекса программных средств идентификации воздействий на файлы

Рассмотрим предложенный комплекс программных средств с помощью IDEF0-схемы, представленной на рисунке 3.11.

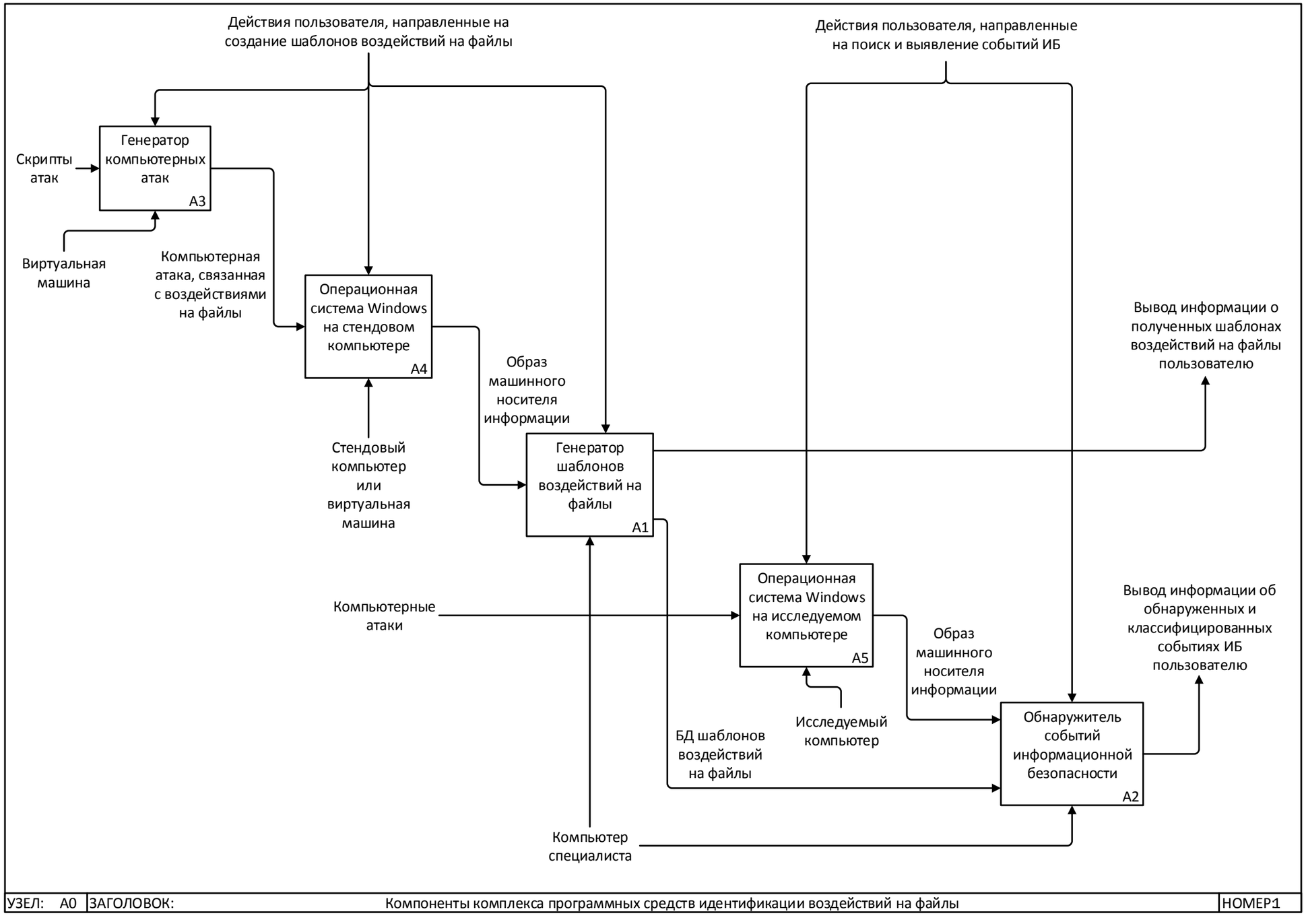


Рисунок 3.11. IDEF0-схема комплекса программных средств

Согласно представленной на рисунке 3.11 схемы, следует выделить два основных этапа использования комплекса программных средств:

1. Подготовка шаблонов воздействий на файлы до инцидента ИБ.

2. Обнаружение и классификация воздействий на файлы с использованием шаблонов для определения событий ИБ.

На этапе подготовки специалист, исходя из используемых технологий и сервисов на исследуемом компьютере (узел А5), готовит его стендовый аналог (узел А4). Реализация КА на ОС и прикладное ПО с целью осуществления воздействий на файлы может происходить с использованием Metasploit (узел А3). Вместе с тем, использование Metasploit не является обязательным: специалист может осуществлять воздействия на файлы с применением интересующего его ПО, выполнять штатные команды ОС в отношении файлов, реализовывать процедуры изменения, копирования, удаления информации посредством файловых операций и т.д.

Осуществив воздействия на файлы, специалист создает образ МНИ стендового компьютера, чтобы исключить искажение информации в процессе анализа. Полученный образ МНИ анализируется на компьютере специалиста с помощью программного средства генерации шаблонов воздействий на файлы (узел А1):

1. Запускается ПО для извлечения потока данных $J из файла журнала изменений тома $UsnJrnl (п. 3.1.1). ПО может быть запущено из интерфейса генератора шаблонов воздействий на файлы.

2. Запускается процесс обработки записей журнала изменений тома $UsnJrnl (п. 3.1.1) для преобразования записей из бинарного вида в таблицу БД формата SQLite. ПО может быть запущено из интерфейса генератора шаблонов воздействий на файлы.

3. Полученная БД открывается в ПО генерации шаблонов воздействий на файлы (п. 3.1.2).

4. Запускается процедура кластеризации записей журнала изменений тома $UsnJrnl с настройками ПО по-умолчанию. Если настройки по-

умолчанию не обеспечивают необходимой точности определения воздействий на файлы, то устанавливаются иные параметры функции плотности распределения вероятности в соответствии со значениями в таблице 2.4, а также параметры временных интервалов $t_1$ и $t_2$ в соответствии с ранее указанными значениями.

5. По окончании кластеризации принимаются или отклоняются предложения по объединению кластеров и полученные воздействия на файлы верифицируются с помощью событийной модели (п. 2.2.6).

6. На основании полученных идентифицированных воздействий на файлы генерируются шаблоны, в которых указываются значимые признаки, устанавливается признак «аномальности» воздействия, определяется приоритет шаблона. Полученные шаблоны сохраняются в БД формата SQLite.

В случае, если БД шаблонов необходимо дополнить, на стендовом компьютере осуществляются новые воздействия на файлы, и процедура генерации повторяется в соответствии с шагами, представленными выше. Для того чтобы не повторять анализ записей предыдущей процедуры получения шаблонов, необходимо с использованием значений $U_f$ и $T_f$ удалить записи в БД SQLite, которые не имеют отношения к новым воздействиям.

Наилучшие результаты обнаружения и классификации событий ИБ достигаются при использовании корректно сформированной БД шаблонов воздействий на файлы. При формировании БД шаблонов следует учитывать несколько рекомендаций:

1. При осуществлении воздействий на файлы необходимо контролировать значение времени ОС стендового компьютера. В дальнейшем это позволит отфильтровать записи журнала, не имеющие отношение к осуществляемым воздействиям на файлы на стендовом компьютере.

2. После извлечения потока $J файла $UsnJrnl и преобразования данных в формат БД SQLite следует удалить записи, которые сохранены в

журнале до осуществления воздействий на файлы. Это действие позволит избежать анализа данных, не имеющих отношения к осуществленным воздействиям.

3. При осуществлении простых воздействий на файлы выполнять действия пользователя с использованием как командной строки, так и графического интерфейса. В случае, если одинаковые действия, осуществлённые из командной строки и графического интерфейса, отличаются, то следует создать шаблон для обоих вариантов действий.

4. При формировании шаблонов воздействий следует проверять предложения программного средства по объединению кластеров, чтобы не упустить из виду сложные комплексные воздействия.

5. При создании шаблонов воздействий, связанных с запуском исполняемых файлов, учитывать, что при запуске ПО создается соответствующий pf файл, который явно указывает на факт запуска. Добавление данных о pf файле в шаблон позволит в дальнейшем обосновано говорить о запуске ПО.

6. При создании шаблонов воздействий, осуществляемых ВПО, следует генерировать шаблон, связанный как с запуском ВПО, так и с файлами, на которые ВПО воздействует. Если ВПО обладает свойствами полиморфизма (изменяет свое содержимое после запуска в целях противодействия сигнатурным методам и средствам обнаружения), то в шаблон запуска следует добавить информацию об изменении содержимого исполняемого файла.

7. При формировании БД шаблонов сохранять только те воздействия, которые имеют отношение к применяемым на целевом компьютере технологиям. Это позволит сократить временные затраты на обнаружение и классификацию воздействий на файлы в рамках расследования инцидента ИБ.

8. При генерации шаблона воздействий на файлы редактировать значения полей «ИФЗ» и «ИРК ФЗ» и отмечать их как значимые только в

случае, если на целевом компьютере значения «ИФЗ» и «ИРК ФЗ» совпадают с таковыми на стендовом, т.е. не изменяются в процессе штатного функционирования ОС на целевом компьютере.

При возникновении инцидента ИБ следует перейти ко второму этапу использования комплекса программных средств. Специалист создает образ МНИ исследуемого компьютера, чтобы исключить искажение информации в процессе анализа. Полученный образ МНИ анализируется на компьютере специалиста с помощью программного средства обнаружения и классификации событий ИБ, использующего шаблоны воздействий на файлы (узел А2):

1. Запускается ПО для извлечения потока данных $J из файла журнала изменений тома $UsnJrnl (п. 3.1.1). ПО может быть запущено из интерфейса ПО обнаружения и классификации воздействии на файлы (п. 3.2.1).

2. Запускается процесс обработки записей журнала изменений тома $UsnJrnl (п. 3.1.1) для преобразования записей из бинарного вида в таблицу БД формата SQLite. ПО может быть запущено из интерфейса ПО обнаружения и классификации воздействии на файлы (п. 3.2.1).

3. Полученная БД и БД шаблонов воздействий открываются в ПО обнаружения и классификации воздействии на файлы.

4. Если необходимо изменить заданные по умолчанию параметры алгоритма поиска и классификации воздействий на файлы с применением БД шаблонов (п. 2.3.3), то в настройках задаются весовые коэффициенты вектора значимых признаков $W$ и устанавливается пороговое значение $G_{threshold}$.

5. При необходимости устанавливаются дополнительные условия «аномальности» в виде предполагаемого начала и конца инцидента ИБ.

6. Запускается процедура обнаружения и классификации воздействий на файлы.

7. По окончании процедуры обнаружения и классификации воздействий на файлы принимаются решения об обнаружении событий ИБ.

С помощью разработанного комплекса программных средств специалист, проводящий расследование инцидента ИБ, может дать ответы на следующие типовые вопросы (перечень не является исчерпывающим):

1. Имеются на исследуемом МНИ следы активности ВПО?
2. Имеются ли на исследуемом МНИ следы запуска ПО, не разрешенного к применению согласно действующей политики безопасности?
3. Имеются ли на исследуемом МНИ следы несанкционированного доступа к информации?
4. Имеются ли на исследуемом МНИ признаки несанкционированного изменения информации?
5. Имеются ли на исследуемом МНИ признаки уничтожения информации?

Рассмотрим применение комплекса программных средств на примере. Пусть на целевом компьютере, работающем по управлением ОС Windows, функционирует сервер общих файлов и папок, а также Web-сервер, который обслуживает Web-приложения, созданные с помощью языка программирования PHP. В исходном коде некоторых приложений присутствуют уязвимости типа «Внедрение кода», которые не были выявлены при проведении аудита ИБ. Чтобы подготовить БД шаблонов воздействий на файлы для использования в случае возникновения инцидента, специалист по ИБ, во избежание внесения искажений в процесс функционирования целевого компьютера создает его копию на базе стендового. Проанализировав результаты аудита ИБ и проведя анализ БД уязвимостей применительно к используемым на целевом компьютере технологиям, специалист готовит шаблоны воздействий на файлы с помощью соответствующего программного средства, описанного ранее. Список подготовленных шаблонов состоит из 52 экземпляров, рассмотрим назначение некоторых из них:

- дополнение и усечение содержимого php файла: для обнаружения попыток внедрения кода в Web-приложения;

- создание файлов с расширением txt: для обнаружения попыток копирования информации;
- переименование файлов — php: для обнаружения попыток переименования текстовых файлов в скрипты php;
- переименование файлов — WNCRY: для обнаружения активности ВПО WannaCry;
- дополнение и усечение содержимого txt файла: для обнаружения попыток переноса текстовой информации в файлы конфигурации сервисов;
- создание файлов с расширением exe: с целью обнаружения попыток загрузки исполняемых файлов на целевой компьютер;
- создание файлов с расширением pf: с целью обнаружения попыток запуска исполняемых файлов на целевом компьютере;
- переименование файлов — exe: для обнаружения попыток запуска приложений.

На рисунке 3.12 представлен пример шаблона «Переименование файлов — php». На указанном примере столбцы таблиц соответствуют следующим полям записей журнала (таблица 1.1) и компонентам вектора *G*:

- столбец «ИФЗ» содержит идентификаторы файловых записей $I_G$;
- столбец «ИРК ФЗ» содержит идентификатор родительских каталогов, принадлежащих $D_G$;
- столбец «Операции» содержит идентификаторы операций, принадлежащих $R_G$;
- столбцы «Имя», «Расширения», «ED», «Шаблоны имен», «Символы имени», содержит имена и их составные элементы, принадлежащие $N_G$;
- столбец «USN» содержит номера записей;
- столбец «Временная отметка» содержит временные отметки создания записей.

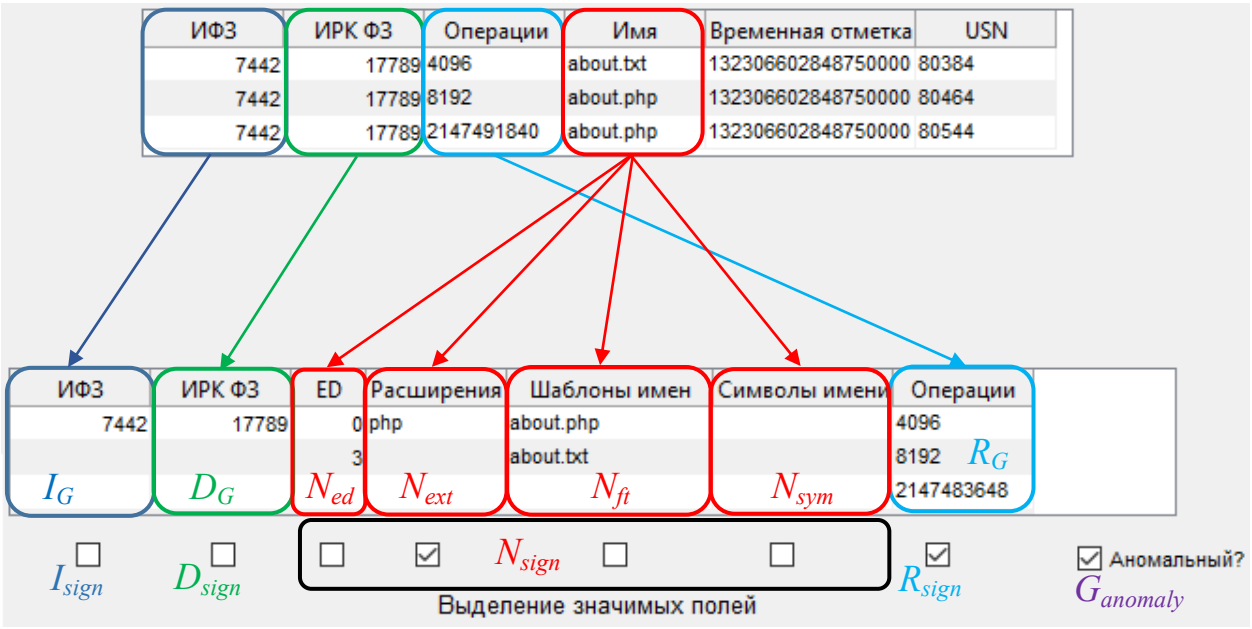


Рисунок 3.12. Пример генерации шаблона воздействия — замена расширения файла с txt на php в целях осуществления атаки типа «Внедрение кода»

Созданная БД шаблонов в последующем будет использована при расследовании инцидента ИБ.

При возникновении инцидента специалист по ИБ создает образ МНИ целевого компьютера, извлекает из образа журнал изменений тома $UsnJrnl и, с применением БД шаблонов, производит обнаружение и классификацию воздействий на файлы. На основании полученных воздействий специалист в соответствии с выражением (2) определяет события ИБ, которые, согласно (1), являются составляющими инцидента ИБ.

Пример обнаруженных и классифицированных воздействий с использованием созданной БД шаблонов представлен на рисунке 3.13. Воздействия, которые не совпали с шаблонами из подготовленной БД, отображены на рисунке 3.14.

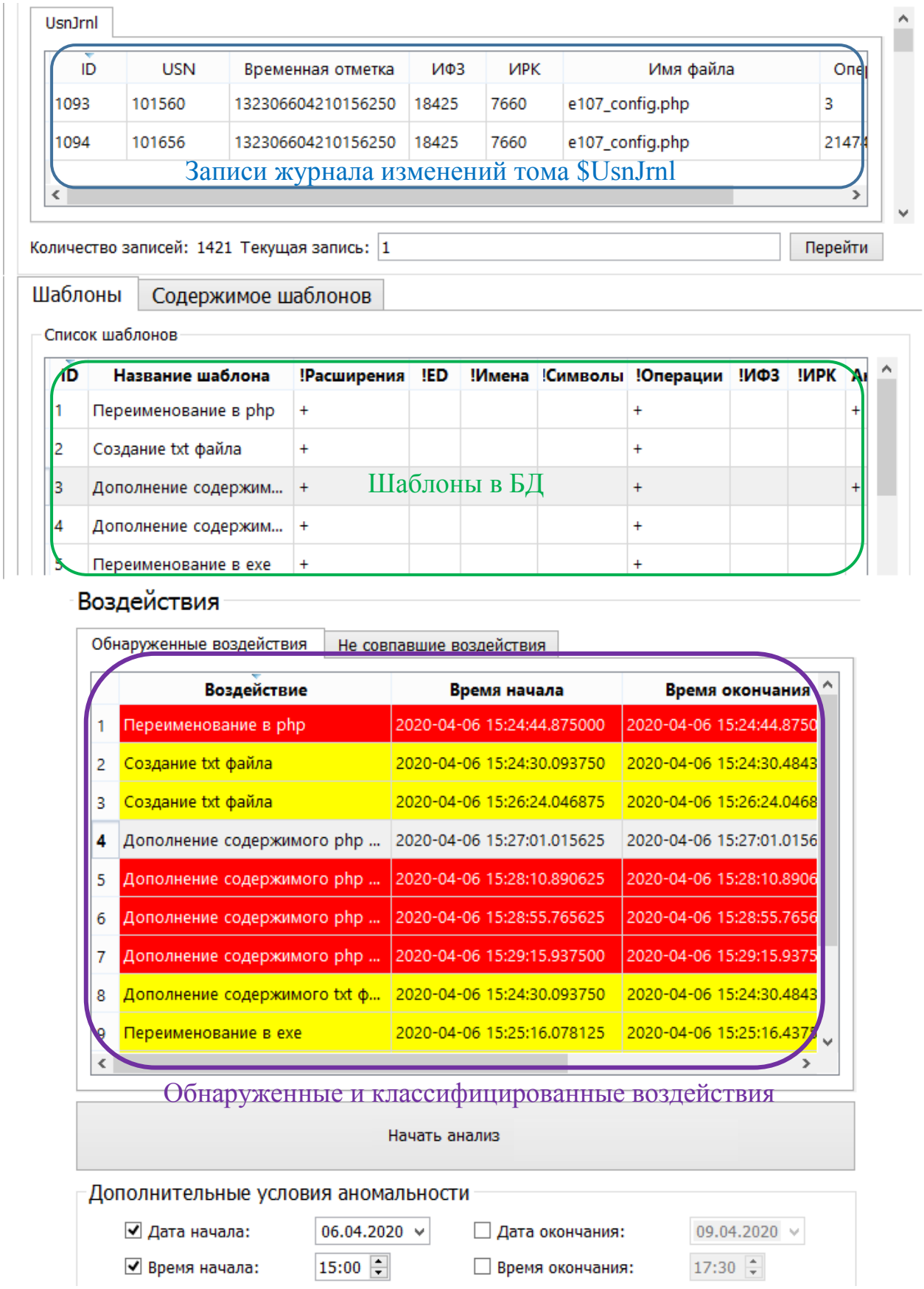


Рисунок 3.13. Пример обнаружения и классификаций воздействий с использованием БД шаблонов

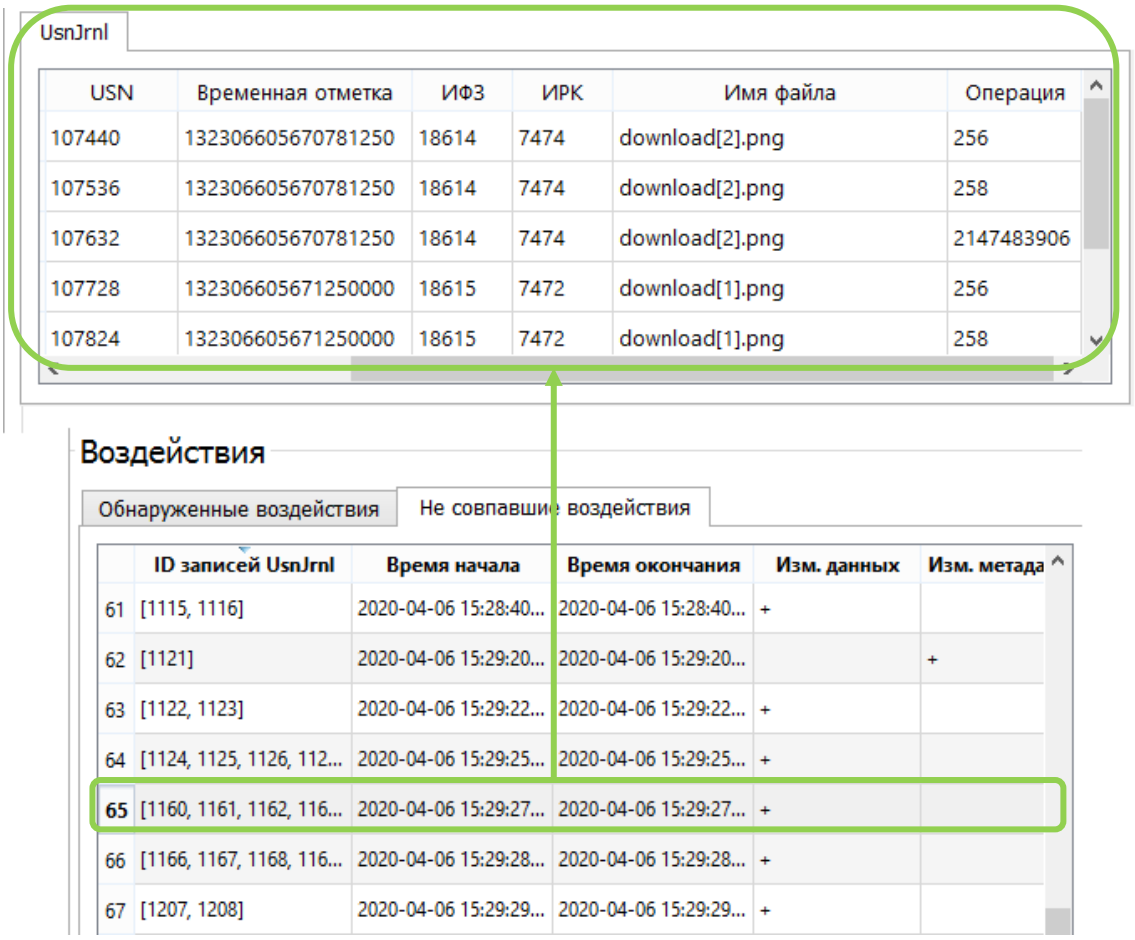


Рисунок 3.14. Воздействия, не совпавшие с шаблонами

Для анализа наличия взаимосвязей между обнаруженными и классифицированными воздействиями на файлы в целях определения событий ИБ удобно использовать графы. Некоторые воздействия на файлы связаны между собой в рамках одного события ИБ, а несколько событий ИБ, составляющих инцидент ИБ, могут быть структурированы и изображены в виде смешанного графа. Вершинами графа являются аномальные и условно аномальные (штриховые линии) воздействия на файлы, ориентированными ребрами являются однозначные и потенциальные (штриховые линии) связи между воздействиями, неориентированными ребрами являются связи между воздействиями для одного файла. Вершины выстроены в порядке осуществления воздействий по времени. Рассмотрим пример проведенного эксперимента по осуществлению атаки на ресурс, направленной на нарушение конфиденциальности и целостности исходного кода Web-приложений.

В процессе расследования инцидента следует сформировать смешанный граф, пример графа представлен на рисунке 3.15.

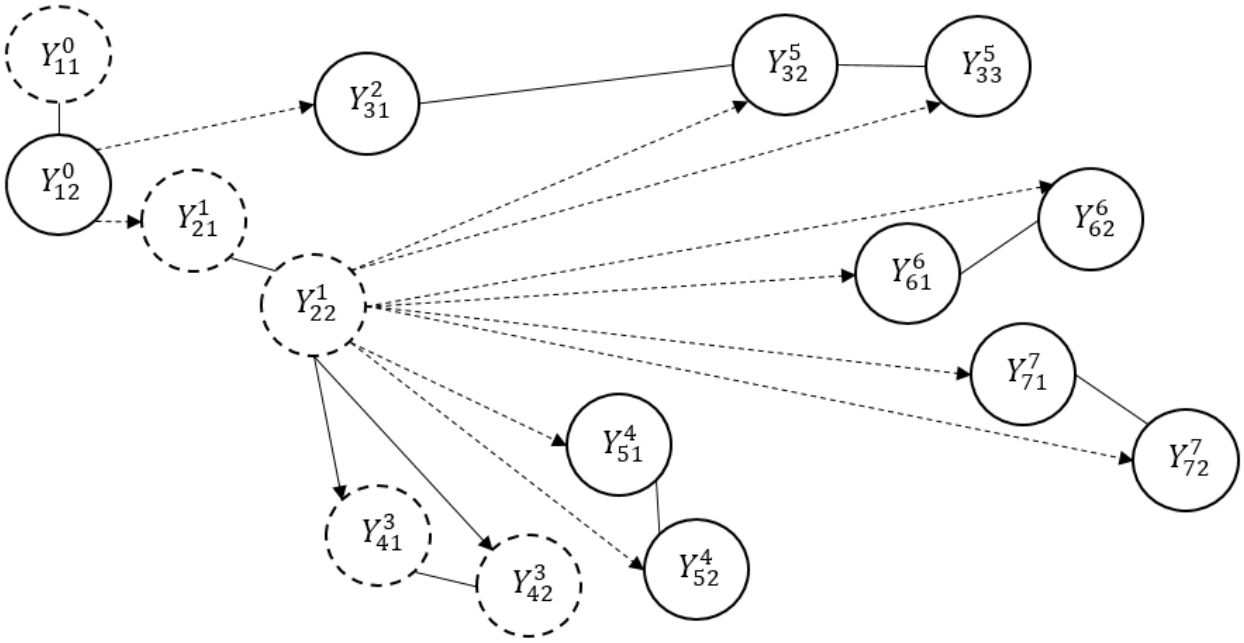


Рисунок 3.15. Пример графа, описывающего инцидент ИБ – нарушение конфиденциальности и целостности исходного кода Web-приложений

На рисунке 3.15 вершины графа характеризуют следующие воздействия:

1. $Y_{11}^0$ – создание файла about.txt в каталоге с файлами Web-ресурса. $Y_{12}^0$ – переименование файла about.txt в about.php. Воздействие $Y_{11}^0$ является условно аномальным, т.к. создание текстовых файлов не всегда свидетельствует о нарушении действующей политики безопасности. $Y_{12}^0$ является аномальным, т.к. about.txt является «Web-шеллом» WSO2 – Web-приложением для получения удаленного контроля над файлами Web-ресурса, и переименование способствует запуску произвольного кода php, содержащегося в этом файле. Событие ИБ, которое характеризуется загрузкой «Web-шелла» на Web-сервер, является аномальным – на Web-сервер не должны быть загружены Web-приложения, не предусмотренные разработчиком Web-ресурса.

2. $Y_{21}^1$ – создание файла tmvwr.jpg в каталоге с файлами Web-ресурса. $Y_{22}^1$ – переименование файла tmvwr.jpg в tmvwr.exe и его открытие (запуск,

т.к. файл исполняемый). Воздействия являются условно аномальными, т.к. по времени находятся после воздействия $Y_{12}^{0}$, из чего можно сделать вывод, что файл tmvwr.jpg мог быть загружен на Web-сервер с помощью WSO2. Событие ИБ является условно аномальным – администратор вправе добавлять исполняемые файлы в каталог Web-ресурса при необходимости, но следует провести динамический анализ файла tmvwr.exe на предмет внедрения ВПО. В случае, если по результатам динамического анализа выяснится, что исполняемый файл является ВПО, то воздействия и событие ИБ следует переквалифицировать в аномальные, а связи между ними — в однозначные.

3. $Y_{31}^{2}$ – воздействие, связанное с редактированием содержимого Web-приложения e107_config.php. Воздействие является аномальным, т.к. в процессе штатного функционирования Web-ресурса исходный код Web-приложений не изменяется. Событие ИБ, связанное с указанным воздействием, является аномальным, т.к. может представлять собой компьютерную атаку типа «внедрение кода».

4. Воздействия $Y_{41}^{3}$ и $Y_{42}^{3}$ – создание и изменение содержимого файла TMVWR.EXE-72508EF8.pf являются условно аномальными, т.к. свидетельствуют о запуске файла tmvwr.exe. Событие ИБ, связанное с воздействиями, является условно аномальным, т.к. запуск ПО может быть предусмотрен алгоритмом работы сервера.

5. Воздействия $Y_{51}^{4}$ и $Y_{52}^{4}$ – создание и изменение содержимого файла @WanaDecryptor@.bmp. Воздействия и событие ИБ являются аномальными, т.к. указанный файл является следствием активности вредоносного программного обеспечения WannaCry.

6. $Y_{32}^{5}$, $Y_{33}^{5}$, $Y_{71}^{7}$, $Y_{72}^{7}$ – воздействия, связанные с редактированием содержимого файлов Web-приложений и их переименованием с добавлением расширения «WNCRY». Воздействия являются аномальными, т.к. в процессе штатного функционирования Web-ресурса исходный код Web-приложений не

изменяется. События ИБ, связанные с указанными воздействиями, являются аномальными, т.к. являются следствием активности ВПО WannaCry.

7. $Y_{61}^{6}$, $Y_{62}^{6}$ – воздействия, связанные с редактированием содержимого файла robots.txt в каталоге Web-ресурса и его переименованием с добавлением расширения «WNCRY». Событие ИБ является аномальным, т.к. является следствием активности ВПО WannaCry.

В указанном примере на графе отображены не все воздействия в силу того, что WannaCry шифрует большее количество файлов в соответствии с алгоритмом своей работы. Тем не менее, в процессе построения графа определяются файлы, которые могут иметь отношение к инциденту, что упрощает его расследование.

Для того чтобы построить граф, подобный представленному на рисунке 3.15, необходимо:

- обнаружить и классифицировать воздействия на файлы;
- отфильтровать нормальные воздействия, не имеющие отношения к инциденту ИБ:
- отсортировать аномальные и условно аномальные воздействия по временной шкале;
- указать взаимосвязи между воздействиями с помощью ребер.

Вершины, связанные неориентированными ребрами, являются частями одного события ИБ. Построение ориентированных ребер графа требует совместного анализа как, например, сработок системы обнаружения атак, так и воздействий на файлы, и носит субъективный характер, но позволяет установить взаимосвязи между событиями ИБ.

## 3.5. Сравнительный анализ предложенных и существующих методов и средств обнаружения и классификации воздействий на файлы

Оценим получаемые результаты при использовании рассмотренных кластеризационного метода идентификации воздействий на файлы и метода экспресс-анализа.

### 3.5.1. Оценка результатов работы кластеризационного метода в сравнении с другими алгоритмами кластеризации

В основе кластеризационного метода лежит алгоритм *k*-средних. Оценка результатов его работы возможна с использованием *F*-меры [91], которая упоминалась в главе 1. Рассмотрим порядок расчета значения *F*-меры.

*F*-мера является гармоническим средним [143] между точностью (Precision) и полнотой (Recall), и рассчитывается по формуле:

$$F-score = \frac{2Precision*Recall}{Precision+Recall} \quad (14)$$

Несмотря на то, что *F*-мера применяется при оценке качества в задаче классификации, определив точность и полноту можно оценить результат кластеризации. Под точностью будем понимать долю точек действительно принадлежащих кластеру относительно всех точек, которые алгоритм отнес к этому кластеру. Полнота – это доля определенных алгоритмом точек, принадлежащих кластеру относительно всех точек этого кластера в наборе данных.

Для того, чтобы рассчитать значение точности и полноты необходимо определить значения *TP*, *TN*, *FP*, *FN*, где:

- *TP* – истинно-положительное решение алгоритма об отнесении точки к кластеру (точка действительно принадлежит кластеру, и алгоритм определил ее в кластер);
- *TN* – истинно-отрицательное решение (точка действительно не принадлежит кластеру, и алгоритм определил ее вне кластера);
- *FP* – ложно-положительное решение (точка действительно не принадлежит кластеру, но алгоритм определил ее в кластер);
- *FN* – ложно-отрицательное решение (точка действительно принадлежит кластеру, но алгоритм определил ее вне кластера);

Тогда точность рассчитывается по формуле:

$$Precision = \frac{TP}{TP+FP} \quad (15)$$

Полнота определяется как:

$$Recall = \frac{TP}{TP+FN} \quad (16)$$

На практике для расчета значений Precision и Recall в случае оценки качества работы алгоритмов классификации используют т.н. матрицу ошибок – квадратную матрицу размером $n*n$, где $n$ – количество классов. Применим этот подход в отношении алгоритма кластеризации. Размерность матрицы определим исходя из количества кластеров $k$. В столбцах матрицы положим ожидаемые специалистом решения относительно принадлежности точек кластерам, в строках – решения алгоритма кластеризации. При рассмотрении каждой точки из набора данных будем инкрементировать элемент матрицы, который стоит на пересечении столбца (решения специалиста) и строки (решения алгоритма). В случае, когда решения специалиста и алгоритма кластеризации полностью совпали, матрица получится диагональной.

Следует отметить, что в ситуации, когда количество точек в наборе данных составляет 500-600 тысяч, а количество кластеров в наборе измеряется тысячами и десятками тысяч, то анализ матрицы ошибок становится затруднительным. Тем не менее, с учетом наличия алгоритма № 1 в структуре кластеризационного метода идентификации воздействий на файлы, исходный набор данных разделяется на блоки, количество точек в которых измеряется десятками, в результате чего размер матрицы ошибок не превышает размера блока. Пример матрицы ошибок представлен на рисунке 3.16.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 4 | 0 | 0 | 0 |
| 3 | 1 | 0 | 6 | 0 | 0 |
| 4 | 0 | 0 | 0 | 2 | 1 |
| 5 | 0 | 0 | 0 | 0 | 5 |

Рисунок 3.16. Пример матрицы ошибок: 20 точек, 5 кластеров

Для расчета промежуточных значений Precision у каждого кластера с использованием матрицы неточностей необходимо вычислить соотношение диагонального элемента к сумме всех элементов в строке с рассматриваемым диагональным элементом. При расчете промежуточных значений Recall вместо суммы по строке используется сумма по столбцу. Итоговые значения Precision и Recall в блоке данных считаются как среднее арифметическое соответствующих промежуточных значений.

Рассмотрим приведенные в таблицах 3.1-3.4 примеры значений Precision, Recall и *F*-score, полученных при использовании нескольких алгоритмов кластеризации на некоторых блоках данных.

Таблица 3.1. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 10 точек при наличии 3 кластеров

| **Алгоритм кластеризации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| *k*-средних | 0,888 | 0,916 | 0,902 |
| *k*-средних++ | 0,888 | 0,916 | 0,902 |
| *k*-медоидов | 0,777 | 0,822 | 0,799 |
| DBSCAN | 0,777 | 0,822 | 0,799 |

Таблица 3.2. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 3 точек при наличии 2 кластеров

| **Алгоритм кластеризации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| *k*-средних | 0,750 | 0,750 | 0,750 |
| *k*-средних++ | 0,750 | 0,750 | 0,750 |
| *k*-медоидов | 0,750 | 0,750 | 0,750 |
| DBSCAN* | 0,250 | 0,250 | 0,250 |

* Примечание: неудовлетворительные по сравнению с другими алгоритмами показатели в данном примере объясняются низкой плотностью распределения точек в пространстве в выбранном блоке при фиксированных значениях $\varepsilon$ и minPts.

Таблица 3.3. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 26 точек при наличии 22 кластеров

| **Алгоритм кластеризации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| *k*-средних | 0,909 | 0,916 | 0,912 |
| *k*-средних++ | 0,909 | 0,916 | 0,912 |
| *k*-медоидов | 0,863 | 0,871 | 0,867 |
| DBSCAN | 0,977 | 0,977 | 0,977 |

Таблица 3.4. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 18 точек при наличии 2 кластеров

| **Алгоритм кластеризации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| *k*-средних | 0,928 | 0,833 | 0,878 |
| *k*-средних++ | 0,928 | 0,833 | 0,878 |
| *k*-медоидов | 0,892 | 0,785 | 0,835 |
| DBSCAN | 0,928 | 0,833 | 0,878 |

Из указанных примеров видно, что представленные алгоритмы дают схожие значениями Precision, Recall и *F*-score, но итоговый результат зависит от выбранных параметров алгоритма и входных данных. В каждом примере путем *ручного* изменения параметров алгоритма можно добиться значений Precision, Recall и *F*-score равными 1. В рассматриваемых же примерах значение количества кластеров *k* определялось автоматически в соответствии с алгоритмом № 2 кластеризационного метода. Для оценки качества кластеризации количество кластеров *k* при работе алгоритмов *k*-средних++ и *k*-медоидов выбиралось аналогично *k*-средних. Автоматический подбор параметров $\varepsilon$ и minPts алгоритма DBSCAN не описан в существующих работах, поэтому были выбраны и зафиксированы такие значения, которые давали приемлемые результаты на большинстве блоков данных.

При определении оптимального *k* в формуле (12) в качестве PDF использовалось нормальное распределение с $\mu = 0$ и $\sigma = 1$. Таблицы с примерами остальных PDF, указанных ранее в таблице 2.4 п. 2.2.4, представлены в приложении В.

Алгоритм № 3 кластеризационного метода идентификации воздействий на файлы применяется для поиска сложных комплексных воздействий на файлы после разбиения набора данных на блоки алгоритмом № 1, но не влияет на значения Precision, Recall и *F*-score при обработке данных с помощью алгоритма *k*-средних.

### 3.5.2. Оценка результатов работы метода экспресс-анализа в сравнении с другими методами классификации

Как упоминалось ранее, *F*-мера применяется при оценке качества

работы классификатора. Сравним метод экспресс-анализа событий с другими популярными методами классификации.

Обучающей выборкой в рамках исследования будем считать набор записей журнала изменений тома $UsnJrnl стендового компьютера. Признаками – значимые компоненты вектора *G*, которые, как уже упоминалось ранее в п. 2.3.1, формируются на основании записей журнала $UsnJrnl, а также $G_{anomaly}$. Таким образом, каждый шаблон можно назвать «описателем» класса. Под выборкой реальных данных будем понимать записи журнала изменений тома $UsnJrnl целевого компьютера. В журнале изменений тома $UsnJrnl целевого компьютера могут встретиться не только классы, представленные шаблонами, но и данные, которые не совпали с шаблонами. Они будут отнесены к специальному классу не совпавших записей.

Сделаем допущение при формировании обучающей выборки – все записи в ней имеют одинаковые типы значимых признаков. Классы, признаки которых отличаются от текущей обучающей выборки, попадут в другую обучающую выборку и будут использованы для обучения следующего классификатора. В результате получится несколько классификаторов, обученных на разном количестве значимых признаков. Без введенного допущения классификатор либо будет учитывать незначимые признаки, что в последующем приведет к неверным результатам классификации на реальных данных в сравнении с алгоритмом предложенного метода экспресс-анализа, который специально разработан для решения конкретной задачи классификации, либо потребует постоянного переобучения в связи с изменением количества значимых признаков, что вызовет затруднения при использовании классификатора в реальной работе.

Сравним результаты классификации воздействий с использованием ансамбля деревьев решений, метода опорных векторов (с линейным ядром) и предложенного метода экспресс-анализа, приведенные в таблицах 3.5-3.7.

Таблица 3.5. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 1000 точек (6 классов) на основании 3 признаков

| **Метод классификации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| Ансамбль деревьев решений | 0,927 | 0,966 | 0,946 |
| Метод опорных векторов | 0,655 | 0,921 | 0,773 |
| Метод экспресс-анализа | 0,912 | 0,960 | 0,935 |

Таблица 3.6. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 100 точек (4 класса) на основании 4 признаков

| **Метод классификации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| Ансамбль деревьев решений | 0,927 | 0,966 | 0,946 |
| Метод опорных векторов | 0,720 | 0,893 | 0,797 |
| Метод экспресс-анализа | 0,923 | 0,966 | 0,944 |

Таблица 3.7. Пример значений Precision, Recall и *F*-score в блоке данных, состоящем из 25 точек (4 класса) на основании 5 признаков

| **Метод классификации** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| Ансамбль деревьев решений | 0,932 | 0,973 | 0,952 |
| Метод опорных векторов | 0,786 | 0,806 | 0,795 |
| Метод экспресс-анализа | 0,957 | 0,960 | 0,958 |

Анализируя представленные в таблицах 3.5-3.7 результаты нетрудно заметить, что предложенный метод экспресс-анализа показывает бóльшую точность классификации, чем метод опорных векторов с линейным ядром, но в некоторых ситуациях уступает ансамблю деревьев решений, что может быть связано с некорректно заданными значениями признаков вектора *G*. Вместе с тем, возможность изменения набора выявляемых классов путем добавления/удаления шаблонов воздействий на файлы позволяет не проводить процедуру обучения классификатора повторно, что является преимуществом предлагаемого метода экспресс-анализа в сравнении с существующими методами классификации.

### 3.5.3. Совместное использование существующего ПО и комплекса программных средств при расследовании инцидента ИБ

Сравнивать указанное в п. 1.6 ПО с разработанным комплексом программных средств в части функциональных возможностей

затруднительно в связи с тем, что рассмотренное ПО решает широкий спектр задач по поиску, извлечению и, отчасти, анализу информации на МНИ и в массивах данных, а комплекс программных средств целенаправленно разрабатывался для автоматизации процесса идентификации воздействий на файлы с целью последующего их обнаружения и классификации. Вместе с тем, сравнение времени работы ПО и комплекса программных средств позволяет оценить возможности их совместного использования. Так, временные затраты на анализ МНИ с помощью Autopsy или Belkasoft Evidence Center зависит от типа МНИ (HDD или SSD), его объема, объема занятого пространства и может составлять от 15 минут до 3-4 часов. Полученный результат требует последующей обработки специалистом для определения событий ИБ в рамках расследования инцидента. ПО LastActivityView извлекает данные из массивов в течение нескольких секунд, но имеет ограниченный набор определяемых действий и не предоставляет возможностей по расширению функционала.

Анализ воздействий на файлы с целью последующего определения событий ИБ с использованием соответствующего ПО из состава комплекса (при подготовленной БД шаблонов) занимает до нескольких минут. В результате такого анализа определяются временной интервал и файлы, которые подверглись воздействиям в рамках инцидента ИБ. В последующем эти данные помогут уменьшить количество критериев поиска информации (сократить количество сигнатур, ключевых слов в зависимости от решаемой задачи поиска) с помощью Autopsy, Belkasoft Evidence Center.

Таким образом, совместное использование комплекса программных средств и рассмотренного ПО позволяет избежать дополнительных временных затрат на исследование всех файлов на МНИ и сконцентрировать внимание специалиста на ликвидации последствий инцидента и выработке рекомендаций по совершенствованию мер, направленных на обеспечение ИБ.

### 3.6. Выводы по главе 3

1. Разработано программное средство генерации шаблонов воздействий на файлы, используемое при расследовании инцидентов ИБ. Работоспособность средства подтверждается вычисленными значениями *F*-меры для использованного алгоритма кластеризации (*F*-мера достигает 91%).

2. Разработано программное средство обнаружения событий ИБ, использующее шаблоны воздействий на файлы. Работоспособность средства подтверждается вычисленными значениями *F*-меры для использованного алгоритма классификации (*F*-мера достигает 95,8%).

3. Разработана методика использования предложенного комплекса программных средств.

4. Проведен сравнительный анализ предложенных и существующих методов и средств обнаружения и классификации воздействий на файлы, результаты которого подтверждают их работоспособность.

# Заключение

В диссертационной работе решена актуальная научно-техническая задача анализа информации о воздействиях на файлы, возникших вследствие инцидента ИБ. Получены следующие основные результаты:

1. Проведен анализ состояния предметной области с указанием недостатков существующих методов, алгоритмов и средств анализа массивов данных, содержащих информацию о воздействиях на файлы.

2. Предложены математические методы и модель, позволяющие в автоматизированном режиме идентифицировать, обнаруживать и классифицировать воздействия на файлы:

   a. впервые обоснована возможность применения математического аппарата сетей Петри для описания процесса идентификации воздействий на файлы, что позволяет формализовать инцидент ИБ как совокупность событий, связанных с воздействиями на файлы;
   b. разработан кластеризационный метод идентификации воздействий на файлы, впервые использующий адаптированные алгоритмы предварительной подготовки входных данных и определения оптимального количества кластеров, позволяющий оптимизировать решение задачи анализа данных о воздействиях на файлы;
   c. разработан метод экспресс-анализа событий ИБ, связанных с воздействиями на файлы, позволяющий выявлять аномальные, условно аномальные и нормальные воздействия на файлы в ходе расследования инцидента ИБ.

3. Разработан комплекс программных средств, позволяющий автоматизировать процесс идентификации, обнаружения и классификации воздействий на файлы в целях определения событий ИБ, лежащих в основе расследуемого инцидента.

Перспективными задачами исследования являются адаптация

предложенной событийной модели процесса идентификации воздействий на файлы в других ФС, модификация предложенных кластеризационного метода идентификации воздействий на файлы и метода экспресс-анализа событий ИБ для обработки записей журналов других ФС, доработка комплекса программных средств для функционирования в других типах ОС.

## Обозначения и сокращения

| | | |
|---|---|---|
| HDD | – | hard disk drive (НЖМД) |
| MDL | – | minimum description length |
| NTFS | – | new technology file system |
| PDF | – | probability density function |
| SSD | – | solid state drive (ТТН) |
| БД | – | база данных |
| ВО | – | временная отметка |
| ВПО | – | вредоносное программное обеспечение |
| ИБ | – | информационная безопасность |
| ИС | – | информационная система |
| ИТ | – | информационные технологии |
| КА | – | компьютерная атака |
| КИИ | – | критическая информационная инфраструктура |
| КС | – | компьютерная система |
| МГК | – | метод главных компонент |
| МНИ | – | машинный носитель информации |
| НЖМД | – | накопитель на жестких магнитных дисках |
| ОС | – | операционная система |
| ПО | – | программное обеспечение |
| РНС | – | рекуррентная нейронная сеть |
| СНС | – | сверточная нейронная сеть |
| ТТН | – | твердотельный накопитель |
| ФС | – | файловая система |

## Список использованных источников

1. О безопасности критической информационной инфраструктуры Российской Федерации: федер. закон от 26 июля 2017 г. №187-ФЗ // Собр. законодательства РФ. — 2017.

2. ГОСТ Р 53114-2008. «Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения.» — М. : ФГУП «Стандартинформ», 2008. — 30 с.

3. Аудит информационной безопасности органов исполнительной власти: учебное пособие для высшего образования / В.Т. Еременко [и др.]. — Орел: ФГБОУ ВО «Орловский государственный университет имени И.С. Тургенева», 2016. — 93 с.

4. Титов С.С., Н.В. Медведев К вопросу об информационной безопасности при делегировании прав // Дискуссия, 2012. — № 8 (26). — С. 111-114.

5. Геут К.Л., Титов С.С. О рекуррентных соотношениях в информационной безопасности // Вестник УрФО. Безопасность в информационной сфере, 2017. — Вып. 23. — № 1. — С. 24-27.

6. Кичигина А.А., Захаров А.А. Оптимизация процесса проверки уязвимостей в информационных системах с помощью вероятностных графов атак // XIII Всероссийская научно-практическая конференция студентов, аспирантов и молодых учёных «Безопасность информационного пространства», 26-28 ноября 2014. — С. 100-104.

7. Соколов А.Н., Лужнов В.С. Специализированные инструменты автоматизированного анализа защищенности информационных систем // Вестник УрФО. Безопасность в информационной сфере, 2020. — Вып. 20. — № 2. — С. 33-38.

8. Богданов В.В., Домуховский А.Н., Лейчук Д.В., Синадский А.Н., Комаров Д.Е. Выявление аномалий в работе информационных систем с помощью машинного обучения // Защита информации. Инсайд, 2020. — № 3(93). — С. 31-35.

9. Духан Е.И., Корольков Ю.Д., Синадский Н.И. Средства защиты информации от несанкционированного доступа: учебное пособие / Е.И. Духан, Ю.Д. Корольков, Н.И. Синадский. — Иркутск: ИГУ, 2012. — 130 с.

10. Девянин П.Н. О проблеме представления формальной модели политики безопасности операционных систем // Труды института системного программирования РАН, 2017. — Т. 29. — № 3. — С. 7-16.

11. Девянин П.Н., Кулямин В.В., Петренко А.К., Хорошилов А.В., Щепетков И.В. Интеграция мандатного и ролевого управления доступом и мандатного контроля целостности в верифицированной иерархической модкели безопасности операционной системы // Труды института системного программирования РАН, 2020. — Т. 32. — № 1. — С. 7-26.

12. Баранкова И.И., Михайлова У.В., Лукьянов Г.И. Подход к проектированию сети предприятия в защищенном исполнении // Вестник УрФО. Безопасность в информационной сфере, 2018. — № 1 (27). — С.24-28.

13. Баранский В.А., Надымова Т.И., Сеньчонок Т.А. Новый алгоритм прохождения графических последовательностей // Сибирские электронные математические известия, 2016. — Т. 13. — С. 269-279.

14. Баранский В.А., Сеньчонок Т.А. О пороговых графах и реализациях графических разбиений // Труды института математики и механики УрО РАН, 2017. — Т. 23. — № 2. — С. 22-31.

15. Гайдамакин Н.А. Мера сходства последовательностей одинаковой размерности // Математические структуры и моделирование, 2016. — № 4 (40). — С. 5-16.

16. Захаров А.А., Захарова И.Г. Дискурсивная метрика в информационном пространстве // Вестник Тюменского государственного университета, 2010. — № 6. — С. 152-156.

17. Попов Е.Ф., Тюкова А.А., Фучко М.М., Захаров А.А. Выявление нетипичных событий средствами статистического анализа // Вестник УрФО.

Безопасность в информационной сфере, 2014. — Вып. 14. — № 4. — С. 24-27.

18. Поршнев С.В., Овечкина Е.В., Каплан В.Е. Теория и алгоритмы аппроксимации эмпирических зависимостей и распределений. — Екатеринбург: УрО РАН, 2006. — 166 с.

19. Ferreira D.R., Vasilyev E. Using logical decision trees to discover the cause of process delays from event logs // Computers in Industry, 2015. — Vol. 70. — pp. 194-207.

20. Sathish V., Sudarsan S.D., Srini R. Event Based Robot Prognostics using Principal Component Analysis // IEEE International Symposium on Software Reliability Engineering Workshops, 2014. — pp. 14-17.

21. Chuah E., Jhumka A., Alt S., Villalobos J.J., Fryman J.B., Barth W.L., Parashar M. Using Resource Use Data and System Logs for HPC System Error Propagation and Recovery Diagnosis // IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking, 2019. — pp. 458-467.

22. Зуев В.Н., Ефимов А.Ю. Нейросетевой поведенческий анализ действий пользователя в целях обнаружения вторжений уровня узла // Программные продукты и системы, 2019. — Т. 32. — № 2. — С. 258-262.

23. Алабугин С.К., Пятницкий И.А., Соколов А.Н. Применение рекуррентных и сверточных нейронных сетей для выявления аномалий технологического процесса // Вестник УрФО. Безопасность в информационной сфере, 2019. — Вып. 32. — № 2. — С. 60-65.

24. Асяев Г.Д., Соколов А.Н. Обнаружение вторжений на основе анализа аномального поведения локальной сети с использованием алгоритмов машинного обучения с учителем // Вестник УрФО. Безопасность в информационной сфере, 2020. — Вып. 35. — № 1. — С. 77-83.

25. Sheluhin O., Osin A. Anomaly States Monitoring of Large-Scale Systems with Intellectual Analysis of System Logs // Proceeding of the 24th conference of FRUCT association, 2019. — pp. 395-401.

26. Шелухин О.И., Рябинин В.С. Обнаружение аномалий больших данных неструктурированных системных журналов // Вопросы кибербезопасности, 2019. — № 2 (30). — С. 36-41.

27. Yuan F., Cao Y., Shang Y., Liu Y., Tan J., Fang B. Insider Threat Detection with Deep Neural Network // Proceedings ICCS 2018, LNCS, 2018. — Vol. 10860. — pp. 43-54.

28. Yen S., Moh M., Moh T.-S. CausalConvLSTM: Semi-Supervised Log Anomaly Detection Through Sequence Modeling // 18th IEEE International Conference on Machine Learning and Applications, 2019. — pp. 1334-1341.

29. Berlin K., Slater D., Saxe J. Malicious Behavior Detection using Windows Audit Logs // Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, 2015. — pp. 35-44.

30. Vaarandi R. A Data Clustering Algorithm for Mining Patterns From Event Logs // Proceedings of the 3rd IEEE Workshop in IP Operations & Management (IPOM), 2003. — pp. 119-126.

31. Vaarandi R., Pihelgas M. LogCluster – A Data Clustering and Pattern Mining Algorithm for Event Logs // 11th International Conference on Network and Service Management (CNSM), 2015. — pp. 1-7.

32. Vaarandi R., Kont M., Pihelgas M. Event Log Analysis with the LogCluster Tool // IEEE Military Communications Conference (MILCOM), 2016. — pp. 982-987.

33. Stein B., Niggemann O. On the Nature of Structure and Its Identification // Lecture Notes in Computer Science, 1999. — Vol. 1665. — pp. 122-134.

34. Studiawan H., Payne C., Sohel F. Graph clustering and anomaly detection of access control log for forensic purposes // Digital Investigation, 2017. — Vol. 21. — pp. 76-87.

35. Studiawan H., Pratomo B.A., Anggoro R. Clustering of SSH Brute-Force Attack Logs Using k-Clique Percolation // International Conference on Information, Communication Technology and System, 2016. — pp. 39-42.

36. ГОСТ Р ИСО/МЭК ТО 18044-2007. «Информационная технология. Методы и средства обеспечения безопасности. Менеджмент инцидентов информационной безопасности.» — М. : ФГУП «Стандартинформ», 2007. — 50 с.

37. Ransomware WannaCry: All you need to know | Kaspersky [Электронный ресурс]. — URL: https://www.kaspersky.com/resource-center/threats/ransomware-wannacry (дата обращения: 20.12.2020).

38. Jigsaw Ransomware | McAfee Research Center [Электронный ресурс]. — URL: https://www.mcafee.com/enterprise/ru-ru/threat-center/threat-landscape-dashboard/ransomware-details.jigsaw-ransomware.html.

39. Стандарт Банка России СТО БР ИББС-1.3-2016 «Обеспечение информационной безопасности организаций банковской системы Российской Федерации. Сбор и анализ технических данных при реагировании на инциденты информационной безопасности при осуществлении переводов денежных средств» [Электронный ресурс]. URL: http://garant.ru/products/ipo/prime/doc/71457690 (дата обращения: 20.12.2020).

40. Кэрриэ Б. Криминалистический анализ файловых систем. — СПб.: Питер, 2007. — 480 с.: ил.

41. Chow K., Law F., Kwan M., Lai K. The Rules of Time on NTFS File System // Second International Workshop on Systematic Approaches to Digital Forensic Engineering. [Электронный ресурс]. — URL : http://i.cs.hku.hk/cisc/forensics/papers/RuleOfTime.pdf (дата обращения: 20.12.2020).

42. Cho G.-S. An Intuitive Computer Forensic Method by Timestamp Changing Patterns // Eight International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. — 2014. — pp.542-548.

43. Матвеева В.С. Криминалистический подход к анализу временных атрибутов файлов в операционной системе семейства Microsoft Windows и файловой системе NTFS // Безопасность информационных технологий. — 2013. — Вып. 1 (20). — С.114-123.

44. Minnaard W. Timestomping NTFS. [Электронный ресурс]. — URL : http://www.delaat.net/rp/2013-2014/p48/report.pdf (дата обращения: 20.12.2020).

45. Knutson T. Filesystem Timestamps: What Makes Them Tick? [Электронный ресурс]. — URL : https://www.sans.org/reading-room/whitepapers/forensics/filesystem-timestamps-tick-36842 (дата обращения: 20.12.2020).

46. Системный вызов – Национальная библиотека им. Н. Э. Баумана [Электронный ресурс]. — URL : https://ru.bsmtu.wiki/Системный_вызов (дата обращения: 20.12.2020)

47. GitHub – slyd0g/TimeStomper: PoC that manipulates Windows file times using SetFileTime() API [Электронный ресурс]. — URL : https://github.com/slyd0g/TimeStomper (дата обращения: 20.12.2020).

48. GitHub – jshicht/Setmace: Manipulate timestamps on NTFS [Электронный ресурс]. — URL : https://github.com/jshicht/Setmace (дата обращения: 20.12.2020).

49. Event Logging (Event Logging) – Win32 apps | Microsoft Docs [Электронный ресурс]. — URL : https://docs.microsoft.com/ru-ru/windows/win32/eventlog/event-logging (дата обращения: 20.12.2020).

50. Reporting Events – Win32 apps | Microsoft Docs [Электронный ресурс]. — URL : https://docs.microsoft.com/ru-ru/windows/win32/eventlog/reporting-an-event (дата обращения: 20.12.2020).

51. Описание событий системы безопасности в Windows 7 и Windows Server 2008 R2. [Электронный ресурс]. — URL: https://support.microsoft.com/ru-ru/help/977519/description-of-security-events-in-windows-7-and-in-windows-server-2008 (дата обращения: 20.12.2020).

52. Xiaoyu H., Shunxiang Wu Vista Event Log File Parsing Based on XML Technology // Proceedings of 4th International Conference on Computer Science & Education, 2009. — pp.1186-1190.

53. Murphey R. Automated Windows event log forensics // Digital Investigation 4S, 2007. — pp. 92-100.

54. Dwyer J., Truta T.M. Finding in Windows Event Logs Using Standard Deviation // 9th IEEE Inernational Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2013. — pp. 563-570.

55. Winzapper [Электронный ресурс]. — URL: http://ntsecurity.nu/toolbox/winzapper (дата обращения: 20.12.2020).

56. Yongjian L., Peng W., Ming X., Ning Z. Automated event log file recovery based on content characters and internal strucuture // The 1st International Conference on Information Science and Engineering (ICISE), 2009. — pp. 4778-4781.

57. [MS-SHLLINK] — v20180912 Shell Link (.LNK) Binary File Format. [Электронный ресурс]. — URL : https://msdn.microsoft.com/en-us/library/dd871305.aspx (дата обращения: 20.12.2020).

58. Parsonage H. The Meaning of Linkfiles In Forensic Examinations. [Электронный ресурс]. — URL : http://computerforensics.parsonage. co.uk/downloads/TheMeaningofLIFE.pdf (дата обращения: 20.12.2020).

59. Carvey H. Jump List Analysis. [Электронный ресурс]. — URL : http://windowsir.blogspot.ru/2011/08/jump-list-analysis.html (дата обращения: 20.12.2020).

60. Carvey H. Jump List Analysis, pt II. [Электронный ресурс]. — URL : http://windowsir.blogspot.ru/2011/08/jump-list-analysis-pt-ii.html (дата обращения: 20.12.2020).

61. Carvey H. Jump List Analysis, pt III. [Электронный ресурс]. — URL : http://windowsir.blogspot.ru/2011/09/jump-list-analysis-pt-iii.html (дата обращения: 20.12.2020).

62. List of Jump List IDs – Forensics Wiki [Электронный ресурс]. — URL : https://forensicswiki.xyz/wiki/index.php?title=List_of_Jump_list_IDs (дата обращения: 20.12.2020).

63. Zimmerman E. Jump lists in depth: Understand the format to better understand what your tools are (or aren't). [Электронный ресурс]. — URL : https://binaryforay.blogspot.ru/2016/02/jump-lists-in-depth-understand-format.html (дата обращения: 20.12.2020).

64. Russinovich M. Inside the Windows Vista kernel: Part 3. Microsoft TechNet Magazine, 2007.

65. Alsulami B., Srinivasan A., Dong H., Mancoridis S. Lightweight Behavioral Malware Detection for Windows Platforms // 12th International Conference on Malicious and Unwanted Software: “Know Your Enemy”, 2017. — pp. 75-81.

66. Uninstall Registry Key. [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/windows/win32/msi/uninstall-registry-key (дата обращения: 20.12.2020).

67. Анонимность работы за компьютером. [Электронный ресурс]. — URL: https://jameszero.net/1849.htm (дата обращения: 20.12.2020).

68. A Quick Glance At The UserAssist Key in Windows. [Электронный ресурс]. — URL: https://windowsexplored.com/2012/02/06/a-quick-glance-at-the-userassist-key-in-windows/ (дата обращения: 20.12.2020).

69. UniLecs #173. Алгоритм ROT13 – Telegraph [Электронный ресурс]. — URL: https://tgraph.io/wiki/UniLecs-173-Algoritm-ROT13-06-06 (дата обращения: 20.12.2020).

70. Cho G.-S., Rogers M. Finding Forensic Information on Creating a Folder in $LogFile of NTFS // Social Informatics and Telecommunications Engineering. — 2012. — pp.211-225.

71. Транзакция – Loginom Wiki [Электронный ресурс]. — URL: https://wiki.loginom.ru/articles/transaction.html (дата обращения: 20.12.2020).

72. Cho G.-S. A computer forensic method for detecting timestamp forgery in NTFS // Computer & Security, 2013. — Vol. 34. — pp.36-46.

73. Palmbach D., Breitinger F. Artifacts for Detecting Timestamp Manipulation in NTFS on Windows and Their Reliability // Forensic Science International: Digital Investigation, 2020. — Vol. 32. — pp.51-59.

74. Fsutil | Microsoft Docs [Электронный ресурс]. — URL: https://docs.microsoft.com/ru-ru/windows-server/administration/windows-commands/fsutil (дата обращения: 20.12.2020).

75. USN_RECORD_V2 – Win32 apps | Microsoft Docs [Электронный ресурс]. — URL: https://docs.microsoft.com/en-us/windows/win32/api/winioctl/ns-winioctl-usn_record_v2 (дата обращения: 20.12.2020).

76. Файловый дескриптор – Национальная библиотека им. Н. Э. Баумана [Электронный ресурс]. — URL: http://ru.bmstu.wiki/Файловый_дескриптор (дата обращения: 20.12.2020).

77. Encrypting File System Technical Reference: Security Policy; Public Key: Security Services [Электронный ресурс]. — URL: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780166(v=ws.10) (дата обращения: 20.12.2020).

78. Advanced $UsnJrnl Forensics: Forensic Insight [Электронный ресурс] — URL : http://forensic-insight.org/wp-content/uploads/2013/C7/F-INSIGHT-Advanced-UsnJrnl-Forensics-English.pdf (дата обращения: 20.12.2020).

79. NTFS Log Tracker – blueangel’s ForensicNote [Электронный ресурс] — URL : https://sites.google.com/site/forensicnote/ntfs-log-tracker (дата обращения: 20.12.2020).

80. Harrell C. Re-Introducing $UsnJrnl [Электронный ресурс] — URL : http://journeyintoir.blogspot.ru/2013/01/re-introducing-usnjrnl.html (дата обращения: 20.12.2020).

81. Uijtewaal F., Prooijen J. UsnJrnl Parsing for File System History [Электронный ресурс] — URL : http://delaat.net/rp/2015-2016/p18/report.pdf (дата обращения: 20.12.2020).

82. Zhuge C., Vaarandi R. Efficient Event Log Mining with LogClusterC // IEEE 3rd International Conference on Big Data Security on Cloud, 2017. — pp. 261-266.

83. RFC 5424 – The Syslog Protocol [Электронный ресурс] — URL : https://tools.ietf.org/html/rfc5424 (дата обращения: 20.12.2020).

84. Vaarandi R., Pihelgas M. LogCluster – A Data Clustering and Pattern Mining Algorithm For Event Logs // 11th International Conference on Network and Service Management (CNSM), 2015. — pp. 31-37.

85. Makanju A., Brooks S., Zincir-Heywood A.N., Milios E.E. LogView: Visualizing Event Log Clusters // Sixth Annual Conference on Privacy, Security and Trust, 2008. — pp. 99-108.

86. Oliner A., Ganapathi A., Xu W. Advances and Challenges in Log Analysis // Communications of the ACM, 2012. — Vol. 55. — No. 2. —pp. 55-61.

87. Alonso J., Belanche L., Avresky D. R. Predicting Software Anomalies using Machine Learning Techniques // IEEE International Symposium on Network Computing and Applications, 2011. — pp. 163-170.

88. He S., Zhu J., He P., Lyu M. R. Experience Report: System Log Analysis for Anomaly Detection // IEEE 27th International Symposium on Software Reliability Engineering, 2016. — pp. 207-218.

89. Грас Дж. Data Science. Наука о данных с нуля: Пер. с англ. — СПб.: БХВ-Петербург, 2019. — 366 с.: ил.

90. Vu Q.H., Ruta D., Cen L. Gradient boosting decision trees for cyber security threats detection based on network events logs // IEEE International Conference on Big Data, 2019. — pp. 5921-5928.

91. C.J. van Rijsbergen: Information Retrieval. 2nd Edition. Butterworth, 1979.

92. Jonathon Shlens. A tutorial on Principal Component Analysis [Электронный ресурс] — URL : https://user.eng.umd.edu/~jzsimon/bio1708L/ref/ShlensPCATutorial.pdf (дата обращения: 20.12.2020).

93. Lalwani H., Gupta R., Srivastava S., Jayaram S. RCA Prediction using Machine Learning // 5th IEEE International WIE Conference on Electrical and Computer Engineering, 2019. — pp. 11-14.

94. Vaarandi R. Mining Event Logs with SLCT and LogHound // Proceedings of the IEEE/IFIP Network Operations and Management Symposium, 2008 — pp. 1071-1074.

95. Галушкин А.И. Нейронные сети: основы теории. — М.: Горячая линия-Телеком, 2014. — 496 с.: ил.

96. Кураева Е.С. Проблемы обучения нейронных сетей / Е.С. Кураева. — Текст : непосредственный // Молодой ученый. — 2020. — № 14 (304). — С. 72-74.

97. Tofallis C. A Better Measure of Relative Prediction Accuracy for Model Selection and Model Estimation // Journal of the Operational Research Society, 2015. — Vol. 66. — pp. 1352-1362.

98. Lu S., Wei X., Li Y., Wang L. Detecting anomaly in Big Data System Logs Using Convolutional Neural Network // IEEE 16th International Conference on Dependable, Autonomic & Secure Comp., 16th International Conference on Pervasive Intelligence & Comp., 4th International Conference on Big Data Intelligence & Comp., and 3rd Cyber Sci.& Tech. Cong., 2018. — pp. 151-158.

99. HDFS Architecture Guide [Электронный ресурс] — URL : https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (дата обращения: 20.12.2020).

100. Ren R., Cheng J., Yin Y., Zhan Y., Wang L., Li J., Luo C. Deep Convolutional Neural Networks for Log Event Classification on Distributed Cluster Systems // IEEE International Conference on Big Data, 2018. — pp. 1639-1646.

101. Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation [Электронный ресурс] — URL : http://deeplearning.net/tutorial/lenet.html (дата обращения: 20.12.2020).

102. Рекуррентные нейронные сети [Электронный ресурс] — URL : https://neerc.ifmo.ru/wiki/index.php?title=Рекуррентные_нейронные_сети (дата обращения: 20.12.2020).

103. The fall of RNN/LSTM. We fell for Recurrent neural network… | by Eugenio Culurciello | Towards Data Science [Электронный ресурс] — URL : https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0 (дата обращения: 20.12.2020).

104. Andrew Ng. CS229 Lecture Notes [Электронный ресурс]. — URL : http://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf (дата обращения: 20.12.2020).

105. Alji M., Chougdali K. Detection of Timestamps Tampering in NTFS using Machine Learning // The international Workshop on Emerging Networks and Communications, Novermber 4-7. — 2019. — pp. 778-784.

106. Qin T., Gao Y., Wei L., Liu Z., Wang C. Potential threats mining based on correlation analysis of multi-type logs // IET Networks, 2018. — Vol. 7. — pp. 299-305.

107. Kaiafas G., Varisteas G., Lagraa S., State R., Nguyen C.D., Ries T., Ourdane M. Detecting Malicious Authentication Events Trustfully // IEEE/IFIP Network Operations and Management Symposium, 2018. — pp. 1-6.

108. Метод опорных векторов (SVM) [Электронный ресурс] — URL : https://neerc.ifmo.ru/wiki/index.php?title=Метод_опорных_векторов_(SVM) (дата обращения: 20.12.2020).

109. Вьюгин В. Математические основы теории машинного обучения и прогнозирования. — МЦМНО, 2013. — 390 с.

110. Jayan K., Rajan A.K. Sys-log Classifier for Complex Event Processing System in Network Security // International Conference on Advances in Computing, Communications and Informatics, 2014. — pp. 2031-2035.

111. Jon Stearley, Adam Oliner. What Supercomputer Say: A Study of Five System Logs. Stanford University Department of Computer Science Palo Ato, CA 94305, 25-28 June 2007.

112. Akanle M., Adetiba E., Akande V., Akinrinmade A., Ajala S., Moninuola F., Badejo J., Adebiyi E. Experimentations with OpenStack System Logs and Support Vector Machine for an Anomaly Detection Model in a Private Cloud Infrastructure // International Conference an Advances in Big Data, Computing and Data Communication Systems (ic ABCD), 2020. — pp. 1-7.

113. Воронцов К.В. Лекции по методу опорных векторов. [Электронный ресурс] — URL : http://ccas.ru/voron/download/SVM.pdf (дата обращения: 20.12.2020).

114. Мандель И. Д. Кластерный анализ. — М.: Финансы и статистика. — 1988. — 176 с: ил.

115. Кузнецов Д.Ю., Трошина Т.Л. Кластерный анализ и его применение // Ярославский педагогический вестник, 2006. — С. 103-107.

116. Makanju A., Zincir-Heywood A.N., Milios E.E. Clustering Event Logs Using Iterative Partitioning // Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009. — pp. 1255-1263.

117. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. — М.: Наука. — 1990. — 384 с.

118. Nagappan M., Vouk M. Abstracting Log Lines to Log Event Types for Mining Software System Logs // 7th IEEE Working Conference on Mining Software Repositories, 2010. — pp. 114-117.

119. Kleinberg J. An Impossibility theorem for clustering // Proceedings of the 15th International Conference on Neural Information Processing Systems, 2002. — pp. 463-470.

120. Lloyd S. P. Least Squares Quantization in PCM // IEEE Transactions on Information Theory, 1982. — Vol. 28. — No. 2. — pp. 129-137.

121. Arthur D., Vassilvitskii S. k-means++: The Advantages of Careful Seeding // Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), 2007. — pp. 1027-1035.

122. Kaufman L., Rousseeuw P.J. Clustering by means of medoids // Statistical Data Analysis Based on the L1-Norm and Related Methods, North-Holland / Elsevier, 1987. — pp. 405-416.

123. Ester M., Kriegel H., Sander J., Xiaowei X. A density-based algorithm for discovering clusters in large spatial databases with noise // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996. — pp. 226-231.

124. Pothen A., Simon H.D., Liou K.-P. Partitioning sparse matrices with eigenvectors of graphs // SIAM journal on Matrix Analysis and Applications, 1990. — Vol. 11. — No. 3 — pp. 430-452.

125. Dhillon I.S., Guan Y., Kulis B. Kernel k-means: spectral clustering and normalized cuts // Proceedings of the tenth ACM International Conference on Knowledge discovery and data mining, 2004. — pp. 551-556.

126. Thorndike R.L. Who belongs in the family? // Psychometrika, 1953. — No. 18. — pp. 267-276.

127. Оценка качества в задаче кластеризации [Электронный ресурс] — URL : https://neerc.ifmo.ru/wiki/index.php?title=Оценка_качества_в_задаче_кластеризации (дата обращения: 20.12.2020).

128. Rissanen J. Modeling by shortest data description // Automatica, 1978. — Vol. 14. — pp. 465-471.

129. Rissanen J. A Universal Prior for Integers and Estimation by Minimum Description Length // The Annals of Statistics, 1983. — Vol. 11. — No. 2. — pp. 416-431.

130. Rissanen J. Universal Coding, Information, Prediction and Estimation // IEEE Transactions on Information Theory, 1984. — Vol. IT-30. — No. 4. — pp. 629-636.

131. Schwarz G.E. Estimating the dimension of a model // Annals of Statistics, 1978. — Vol. 6. — No. 2 — pp. 461-464.

132. Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб. для вузов — 3-е изд., перераб. и доп. — М.: Высш. шк., 2001. — 343 с.: ил.

133. Питерсон Дж. Теория сетей Петри и моделирование систем: пер. с англ. — М.: Мир, 1984. — 264 с. — ил.

134. Effendi Y.A., Sarno R. Discovering Process Model from Event Logs by Considering Overlapping Rules // 4th International Conference on Electrical Engineering, Computer Science and Informatics, 2017. — pp.1-6.

135. GitHub — sarahtattersall/PIPE: PIPE – Platform Independent Petri Net Editor [Электронный ресурс]. — URL : https://github.com/sarahtattersall/PIPE (дата обращения: 20.12.2020).

136. Dingle N., Knottenbelt W., Suto T. PIPE2: A tool for the Performance Evaluation of Generalized Stochastic Petri Nets // ACM SIGMETRICS Performance Evaluation Review (Special Issue on Tools for Computer Performance Modeling and Reliability Analysis), 2009. — № 36. — pp. 34-39.

137. Using minimum description length to optimize the ‘k’ in k-medoids [Электронный ресурс]. — URL: http://erikerlandson.github.io/blog/2016/08/03/x-medoids-using-minimum-description-length-to-identify-the-k-in-k-medoids (дата обращения: 20.12.2020).

138. Ukkonen E. Approximate string-matching with q-grams and maximal matches // Theoretical Computer Science, 1992. — pp. 191-211.

139. Navarro G. A Guided Tour to Approximate String Matching // ACM Computting Surveys, 2001. — Vol. 33. — No. 1. — pp. 31-88.

140. GitHub – jschicht/ExtractUsnJrnl: Tool to extract the $UsnJrnl from an NTFS volume [Электронный ресурс]. — URL: https://github.com/jschicht/ExtractUsnJrnl (дата обращения: 20.12.2020).

141. SQLite Home Page [Электронный ресурс]. — URL: https://www.sqlite.org/index.html (дата обращения: 20.12.2020).

142. Р 50.1.028-2001 «Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования» — М. : «ВНИИстандарт», 2002. — 78 с.

143. Harmonic Mean – from Wolfram MathWorld [Электронный ресурс]. — URL: https://mathworld.wolfram.com/HarmonicMean.html (дата обращения: 20.12.2020).

**Список публикаций автора по теме диссертации**

**Статьи, опубликованные в рецензируемых научных журналах и изданиях, определенных ВАК РФ и Аттестационным советом УрФУ:**

144. Гайдамакин Н.А. Метод экспресс-анализа событий, связанных с воздействиями на файлы, предназначенный для расследования инцидентов информационной безопасности / Н.А. Гайдамакин, **Р.В. Гибилинда**, Н.И. Синадский // Вестник СибГУТИ. — 2020. — № 4. — С.3-13. (0,68 п.л. / 0,23 п.л.).

145. Gaidamakin N.A. File Operations Information Collecting Software Package Used in the Information Security Incidents Investigation / Nikolay A Gaidamakin, **Roman V Gibilinda** and Nikolay I Sinadskiy // 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). – 2020. – № 9117671. – pp. 559-562. (0,25 п.л. / 0,08 п.л.) (Scopus).

146. Гайдамакин Н.А. Событийная модель процесса идентификации воздействий на файлы при расследовании инцидентов информационной безопасности, основанная на математическом аппарате сетей Петри / Н.А. Гайдамакин, **Р.В. Гибилинда**, Н.И. Синадский // Вестник СибГУТИ. — 2020. — № 1. — С.73-88. (1 п.л. / 0,33 п.л.).

147. **Гибилинда Р.В.** Кластеризационный метод идентификации воздействий на файлы с применением алгоритма *k*-средних, используемый при расследовании инцидентов информационной безопасности / **Р.В. Гибилинда** // Вестник УрФО. Безопасность в информационной сфере. — 2020. — Вып. 35 — № 1. — С. 35-47. (0,81 п.л.).

**Патенты и программы**:

148. Свидетельство № 2020616110 Российская Федерация. Свидетельство о государственной регистрации программы для ЭВМ

«Программное средство обнаружения событий информационной безопасности, использующее шаблоны воздействий на файлы» / **Р.В. Гибилинда**. – Заявка № 2020613614 от 26.03.2020; дата гос. регистрации в Реестре 09.06.2020. – Реестр программ для ЭВМ. – 1 с.

149. Свидетельство № 2020616109 Российская Федерация. Свидетельство о государственной регистрации программы для ЭВМ «Программное средство генерации шаблонов воздействий на файлы, используемое при расследовании инцидентов информационной безопасности» / **Р.В. Гибилинда**. – Заявка № 2020613613 от 26.03.2020; дата гос. регистрации в Реестре 09.06.2020. – Реестр программ для ЭВМ. – 1 с.

**Другие публикации**:

150. **Гибилинда Р.В.** Автоматизация процесса идентификации воздействий на файлы с применением кластеризационного метода при расследовании инцидентов информационной безопасности / **Гибилинда Р.В.**, Синадский Н.И. // II Всероссийская научная конференция (с приглашением зарубежных ученых) «Фундаментальные проблемы информационной безопасности в условиях цифровой трансформации» (FISP-2020). — 2020. — С. 284-287 (0,31 п.л. / 0,16 п.л.).

151. **Гибилинда Р.В.** Идентификация воздействий на файлы и верификация массивов данных, содержащих информацию о воздействиях, при расследовании инцидентов информационной безопасности / **Гибилинда Р.В.**, Синадский Н.И. // Современные проблемы радиоэлектроники и телекоммуникаций: сб. науч. тр. / под ред. Ю. Б. Гимпилевича. — Москва-Севастополь: Изд-ва: РНТОРЭС им. А.С. Попова, СевГУ. — 2020. — № 3. — С. 219-219 (0,06 п.л. / 0,03 п.л.).

# Приложение А. Акты о внедрении

**АО «Концерн ВКО «Алмаз – Антей»**

Акционерное общество
**"Опытное конструкторское бюро**
**"Новатор"**
**(АО «ОКБ «Новатор»)**
620017, г. Екатеринбург, пр. Космонавтов, 18
Тел.: (343) 264-10-00
Факс: (343) 264-13-00, (343) 331-17-73
E-mail: main@okb-novator.ru
Для телеграмм "Лен"

«_____» _____ 20___ г. № _____

На Ваш № _____ от «___» _____20____г.

УТВЕРЖДАЮ

Заместитель генерального директора
по режиму и безопасности
АО «ОКБ «Новатор»

Р.Р. Останин

23. 12 2020 г.

АКТ ВНЕДРЕНИЯ

комплекса программных средств идентификации воздействий
на файлы при расследовании инцидентов информационной безопасности

С целью усовершенствования способов расследования инцидентов информационной безопасности, возникающих в процессе эксплуатации автоматизированных рабочих мест сотрудников в 2020 году в акционерном обществе «Опытное конструкторское бюро «Новатор» внедрен комплекс программных средств идентификации воздействий на файлы, разработанный Р.В. Гибилиндой и к.т.н., доцентом Н.И. Синадским.

Комплекс программных средств расширил возможности специалистов, ответственных за обеспечение информационной безопасности, в части выявления попыток несанкционированного доступа к информации и поиска следов запуска стороннего программного обеспечения, в том числе вредоносного.

Внедрение указанного комплекса позволило дополнить принятые организационно-технические меры по обеспечению информационной безопасности.

Руководитель направления С.В. Леонтьев

Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина» (УрФУ)

ул. Мира, 19, Екатеринбург, 620002,
факс: +7 (343) 375-97-78; тел.: +7 (343) 374-38-84
контакт-центр: +7 (343) 375-44-44, 8-800-100-50-44 (звонок бесплатный)
e-mail: rector@urfu.ru, www.urfu.ru
ОКПО 02069208, ОГРН 1026604939855, ИНН/КПП 6660003190/667001001

10 МАР 2021 № 01.09 - 07/134

На № _____ от _____

УТВЕРЖДАЮ
Проректор по науке
Германенко А.В. _____
«___» _____ 2021 года

# АКТ

## Об использовании результатов диссертационного исследования Гибилинды Романа Владимировича

Мы, нижеподписавшиеся, представители Уральского Федерального университета имени первого Президента России Б.Н. Ельцина (УрФУ) директор Института Радиоэлектроники и Информационных технологий–РТФ (ИРИТ-РтФ) Илья Николаевич Обабков и директор учебно-научного центра Информационная безопасность (УНЦ ИБ) Поршнев Сергей Владимирович составили настоящий акт в том, что результаты диссертационного исследования Гибилинды Р.В. используются при реализации учебного процесса по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности» в том числе:

- метод экспресс-анализа событий информационной безопасности, связанных с воздействиями на файлы, используется в лабораторных работах по расследованию инцидентов в компьютерных системах.

Директор ИРИТ-РтФ _____ И.Н. Обабков

Директор УНЦ ИБ _____ С.В. Поршнев

УТВЕРЖДАЮ

Генеральный директор ООО «УЦСБ»
к.т.н. В.В. Богданов

____._____ 2021 г.

**АКТ ВНЕДРЕНИЯ**

**Комплекса программных средств идентификации воздействий на файлы при расследовании инцидентов информационной безопасности**

г. Екатеринбург ____._____ 2021 г.

Комплекс программных средств идентификации воздействий на файлы, разработанный Р.В. Гибилиндой и к.т.н., доцентом Н.И. Синадским, был внедрен в 2020 году в обществе с ограниченной ответственностью «Уральский центр систем безопасности» в режиме опытной эксплуатации с целью использования комплекса для расследования инцидентов информационной безопасности, возникающих, в том числе, в процессе эксплуатации автоматизированных систем управления технологическими процессами.

Применение комплекса программных средств позволило восстановить ход инцидента (определить файлы, которые подвергались воздействиям и установить временной интервал возникновения инцидента), сократив временные затраты аналитика на проведение расследования инцидента информационной безопасности до 20 минут.

Использование комплекса значительно ускоряет проведение мероприятий по реагированию / расследованию инцидентов информационной безопасности, возникающих в, в том числе, в информационной инфраструктуре промышленных предприятий.

Директор Корпоративного центра мониторинга ИБ средств и систем информатизации ООО «УЦСБ» Амиров Р.М.

# Приложение Б. Таблицы позиций и переходов сети Петри, моделирующей процесс идентификации воздействий на файлы

Таблица Б.1. Описание позиций

| **№ позиции** | **Описание** |
|---|---|
| Позиции маркировки, определяющие начальное состояние сети Петри | |
| $P_1$ | Добавление файла с признаками $I_{f1}$, $D_{f1}$, $N_{f1}$ на вход сети Петри |
| $P_2$ | Добавление файла с признаками $I_{f2}$, $D_{f2}$, $N_{f2}$ на вход сети Петри |
| $P_3$ | Условие равенства $I_{f1}$ и $I_{f2}$. Установка фишки, если $I_{f1} = I_{f2}$ |
| $P_4$ | Условие равенства $D_{f1}$ и $D_{f2}$. Установка фишки, если $D_{f1} = D_{f2}$ |
| $P_5$ | Условие равенства $N_{f1}$ и $N_{f2}$. Установка фишки, если $N_{f1} = N_{f2}$ |
| $P_6$ | Признак изменения содержимого файла. Установка фишки необходима, если содержимое файла изменялось |
| $P_{37}$ | Имитация задержки осуществления файловой операции. Установка фишки необходима для запуска перехода $T_{25}$ |
| Вспомогательные и алгоритмические позиции | |
| $P_7$ | Файл 1 существует |
| $P_8$ | Файл 2 существует |
| $P_9$ | $I_{f1} \notin \varnothing$ |
| $P_{10}$ | $I_{f2} \notin \varnothing$ |
| $P_{11}$ | $D_{f1} \notin \varnothing$ |
| $P_{12}$ | $D_{f2} \notin \varnothing$ |
| $P_{13}$ | $N_{f1} \notin \varnothing$ |
| $P_{14}$ | $N_{f2} \notin \varnothing$ |
| $P_{15}$ | $I_{f1}, I_{f2} \notin \varnothing$ |
| $P_{16}$ | $D_{f1}, D_{f2} \notin \varnothing$ |
| $P_{17}$ | $N_{f1}, N_{f2} \notin \varnothing$ |
| $P_{18}$ | Файлы 1 и 2 существуют |
| $P_{19}$ | $I_{f1} = I_{f2}$ |
| $P_{20}$ | $I_{f1} \neq I_{f2}$ |
| $P_{21}$ | $D_{f1} = D_{f2}$ |

| № позиции | Описание |
|---|---|
| $P_{22}$ | $D_{f1} \neq D_{f2}$ |
| $P_{23}$ | $N_{f1} = N_{f2}$ |
| $P_{24}$ | $N_{f1} \neq N_{f2}$ |
| $P_{25}$ | $I_{f1}, I_{f2} \notin \varnothing$ и $I_{f1} = I_{f2}$ |
| $P_{26}$ | $I_{f1}, I_{f2} \notin \varnothing$ и $I_{f1} \neq I_{f2}$ |
| $P_{27}$ | $I_{f1}, I_{f2} \notin \varnothing$, $I_{f1} \neq I_{f2}$, $N_{f1} \neq N_{f2}$ |
| $P_{28}$ | $I_{f1}, I_{f2} \notin \varnothing$, $I_{f1} \neq I_{f2}$, $N_{f1} = N_{f2}$, $D_{f1} = D_{f2}$ |
| $P_{38}$ | Имитация завершения файловой операции, после задержки в $T_{25}$ |
| Позиции маркировки, указывающие на классифицированную файловую операцию | |
| $P_{29}$ | Создание файла (таблица 2.1, $O1$) |
| $P_{30}$ | Удаление файла (таблица 2.1, $O2$) |
| $P_{31}$ | Переименование файла (таблица 2.1, $O3$) |
| $P_{32}$ | Изменение служебной информации файла (таблица 2.1, $O5$) |
| $P_{33}$ | Изменение содержимого файла (таблица 2.1, $O4$) |
| $P_{34}$ | Перемещение файла в пределах логического раздела (таблица 2.1, $O6$) |
| $P_{35}$ | Копирование файла в пределах каталога (таблица 2.1, $O7$) |
| $P_{36}$ | Копирование файла в другой каталог (таблица 2.1, $O8$) |

Таблица Б.2. Описание переходов

| № перехода | Описание |
|---|---|
| $T_1$ | Получение информации о файле с признаками $I_{f1}$, $D_{f1}$, $N_{f1}$ |
| $T_2$ | Получение информации о файле с признаками $I_{f2}$, $D_{f2}$, $N_{f2}$ |
| Вспомогательные и алгоритмические переходы | |
| $T_3$ | Получение значения – равенство $I_{f1}$ и $I_{f2}$ |
| $T_4$ | Получение значения – неравенство $I_{f1}$ и $I_{f2}$ |
| $T_5$ | Получение значения – равенство $D_{f1}$ и $D_{f2}$ |
| $T_6$ | Получение значения – неравенство $D_{f1}$ и $D_{f2}$ |
| $T_7$ | Получение значения – равенство $N_{f1}$ и $N_{f2}$ |
| $T_8$ | Получение значения – неравенство $N_{f1}$ и $N_{f2}$ |
| $T_9$ | Исключение операций «Создание файла» и «Удаление файла» |
| $T_{10}$ | Определение принадлежности $I_{f1}$ и $I_{f2}$ к непустому множеству |

| № перехода | Описание |
|---|---|
| $T_{11}$ | Определение принадлежности $D_{f1}$ и $D_{f2}$ к непустому множеству |
| $T_{12}$ | Определение принадлежности $N_{f1}$ и $N_{f2}$ к непустому множеству |
| $T_{13}$ | Переход к файловым операциям с равными признаками $I_{f1}$ и $I_{f2}$ |
| $T_{14}$ | Переход к файловым операциям с неравными признаками $I_{f1}$ и $I_{f2}$ |
| $T_{15}$ | Переход к файловым операциям, где $I_{f1} = I_{f2}$, $N_{f1} = N_{f2}$ |
| $T_{16}$ | Переход к файловым операциям, где $I_{f1} = I_{f2}$, $N_{f1} = N_{f2}$, $D_{f1} = D_{f2}$ |
| $T_{25}$ | Не примитивный переход, имитирующий время выполнения файловым операциям |
| Переходы, идентифицирующие файловые операции | |
| $T_{17}$ | Создание файла (таблица 2.1, $O1$) |
| $T_{18}$ | Удаление файла (таблица 2.1, $O2$) |
| $T_{19}$ | Переименование файла (таблица 2.1, $O3$) |
| $T_{20}$ | Изменение служебной информации файла (таблица 2.1, $O5$) |
| $T_{21}$ | Изменение содержимого файла (таблица 2.1, $O4$) |
| $T_{22}$ | Перемещение файла в пределах одного логического раздела (таблица 2.1, $O6$) |
| $T_{23}$ | Копирование файла в рамках каталога (таблица 2.1, $O7$) |
| $T_{24}$ | Копирование файла в другой каталог (таблица 2.1, $O8$) |

# Приложение В. Результаты сравнительного анализа работы алгоритма *k*-средних при разных плотностях распределения вероятностей и их параметрах

Примечание: курсивом выделены те значения Precision, Recall и *F*-Score, которые не удовлетворяют потребностям специалиста при проведении кластеризации.

Таблица В.1. Расчет Precision, Recall и *F*-Score для применяемого в качестве PDF нормального распределения в случае 2 кластеров, состоящих из 18 точек

| **Значение параметров PDF** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| $\mu = 0;\ \sigma = 0{,}4$ | *0,946* | *0,678* | *0,790* |
| $\mu = 1;\ \sigma = 0{,}4$ | *0,946* | *0,678* | *0,790* |
| $\mu = 2;\ \sigma = 0{,}4$ | *0,946* | *0,678* | *0,790* |
| $\mu = 0;\ \sigma = 0{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 1;\ \sigma = 0{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 2;\ \sigma = 0{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 0;\ \sigma = 1$ | 0,928 | 0,833 | 0,878 |
| $\mu = 1;\ \sigma = 1$ | 0,928 | 0,833 | 0,878 |
| $\mu = 2;\ \sigma = 1$ | 0,928 | 0,833 | 0,878 |
| $\mu = 0;\ \sigma = 1{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 1;\ \sigma = 1{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 2;\ \sigma = 1{,}5$ | 0,928 | 0,833 | 0,878 |
| $\mu = 0;\ \sigma = 1{,}6$ | *0,916* | *0,642* | *0,755* |
| $\mu = 1;\ \sigma = 1{,}6$ | *0,916* | *0,642* | *0,755* |
| $\mu = 2;\ \sigma = 1{,}6$ | *0,916* | *0,642* | *0,755* |

Таблица В.2. Расчет Precision, Recall и *F*-Score для применяемого в качестве PDF гамма-распределения в случае 2 кластеров, состоящих из 18 точек

| **Значение параметров PDF** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| $\alpha = 0{,}5;\ \beta = 0{,}7$ | *0,846* | *0,678* | *0,753* |
| $\alpha = 1;\ \beta = 0{,}7$ | *0,846* | *0,678* | *0,753* |
| $\alpha = 2;\ \beta = 0{,}7$ | *0,846* | *0,678* | *0,753* |
| $\alpha = 0{,}5;\ \beta = 0{,}8$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 1;\ \beta = 0{,}8$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 2;\ \beta = 0{,}8$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 0{,}5;\ \beta = 0{,}9$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 1;\ \beta = 0{,}9$ | 0,928 | 0,833 | 0,878 |

| | | | |
|---|---|---|---|
| $\alpha = 2;\ \beta = 0{,}9$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 0{,}5;\ \beta = 1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 1;\ \beta = 1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 2;\ \beta = 1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 0{,}5;\ \beta = 1{,}1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 1;\ \beta = 1{,}1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 2;\ \beta = 1{,}1$ | 0,928 | 0,833 | 0,878 |
| $\alpha = 0{,}5;\ \beta = 1{,}2$ | *0,907* | *0,678* | *0,776* |
| $\alpha = 1;\ \beta = 1{,}2$ | *0,907* | *0,678* | *0,776* |
| $\alpha = 2;\ \beta = 1{,}2$ | *0,907* | *0,678* | *0,776* |

Таблица В.3. Расчет Precision, Recall и *F*-Score для применяемого в качестве PDF распределения Стьюдента в случае 2 кластеров, состоящих из 18 точек

| **Значение параметров PDF** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| $\nu = 1$ | 0,928 | 0,806 | 0,863 |
| $\nu = 2$ | 0,928 | 0,833 | 0,878 |
| $\nu = 3$ | 0,928 | 0,833 | 0,878 |
| $\nu = 4$ | 0,928 | 0,833 | 0,878 |
| $\nu = 5$ | 0,928 | 0,833 | 0,878 |
| $\nu = 6$ | *0,846* | *0,678* | *0,753* |

Таблица В.4. Расчет Precision, Recall и *F*-Score для применяемого в качестве PDF распределения $\chi^2$ в случае 2 кластеров, состоящих из 18 точек

| **Значение параметров PDF** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| $\nu = 1$ | 0,906 | 0,833 | 0,868 |
| $\nu = 2$ | 0,928 | 0,833 | 0,878 |
| $\nu = 3$ | 0,928 | 0,833 | 0,878 |
| $\nu = 4$ | 0,928 | 0,833 | 0,878 |
| $\nu = 5$ | 0,928 | 0,833 | 0,878 |
| $\nu = 6$ | *0,846* | *0,678* | *0,753* |

Таблица В.5. Расчет Precision, Recall и *F*-Score для применяемого в качестве PDF распределения Фишера в случае 2 кластеров, состоящих из 18 точек

| **Значение параметров PDF** | **Precision** | **Recall** | ***F*-score** |
|---|---|---|---|
| $\nu 1 = 1;\ \nu 2 = 1$ | 0,912 | 0,807 | 0,856 |
| $\nu 1 = 2;\ \nu 2 = 1$ | 0,906 | 0,807 | 0,854 |
| $\nu 1 = 1;\ \nu 2 = 2$ | 0,916 | 0,833 | 0,873 |
| $\nu 1 = 2;\ \nu 2 = 2$ | 0,916 | 0,833 | 0,873 |
| $\nu 1 = 1;\ \nu 2 = 3$ | 0,912 | 0,817 | 0,862 |

| | | | |
|---|---|---|---|
| $v1 = 2;\ v2 = 3$ | 0,912 | 0,817 | 0,862 |
| $v1 = 1;\ v2 = 4$ | 0,912 | 0,817 | 0,862 |
| $v1 = 2;\ v2 = 4$ | 0,912 | 0,817 | 0,862 |
| $v1 = 1;\ v2 = 5$ | 0,912 | 0,817 | 0,862 |
| $v1 = 2;\ v2 = 5$ | 0,912 | 0,817 | 0,862 |
| $v1 = 1;\ v2 = 6$ | *0,876* | *0,678* | *0,764* |
| $v1 = 2;\ v2 = 6$ | *0,876* | *0,678* | *0,764* |

# Приложение Г. Свидетельства о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

## СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

**№ 2020616109**

**«Программное средство генерации шаблонов воздействий на файлы, используемое при расследовании инцидентов информационной безопасности»**

Правообладатель: ***Гибилинда Роман Владимирович (RU)***

Автор: ***Гибилинда Роман Владимирович (RU)***

Заявка № **2020613613**
Дата поступления **26 марта 2020 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***09 июня 2020 г.***

*Руководитель Федеральной службы по интеллектуальной собственности*

*Г.П. Ивлиев*

РОССИЙСКАЯ ФЕДЕРАЦИЯ

# СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

**№ 2020616110**

**«Программное средство обнаружения событий информационной безопасности, использующее шаблоны воздействий на файлы»**

Правообладатель: ***Гибилинда Роман Владимирович (RU)***

Автор: ***Гибилинда Роман Владимирович (RU)***

Заявка № **2020613614**
Дата поступления **26 марта 2020 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***09 июня 2020 г.***

*Руководитель Федеральной службы*
*по интеллектуальной собственности*

*Г.П. Ивлиев*

# Приложение Д. Пример выполнения сети Петри для идентификации простого воздействия на файл

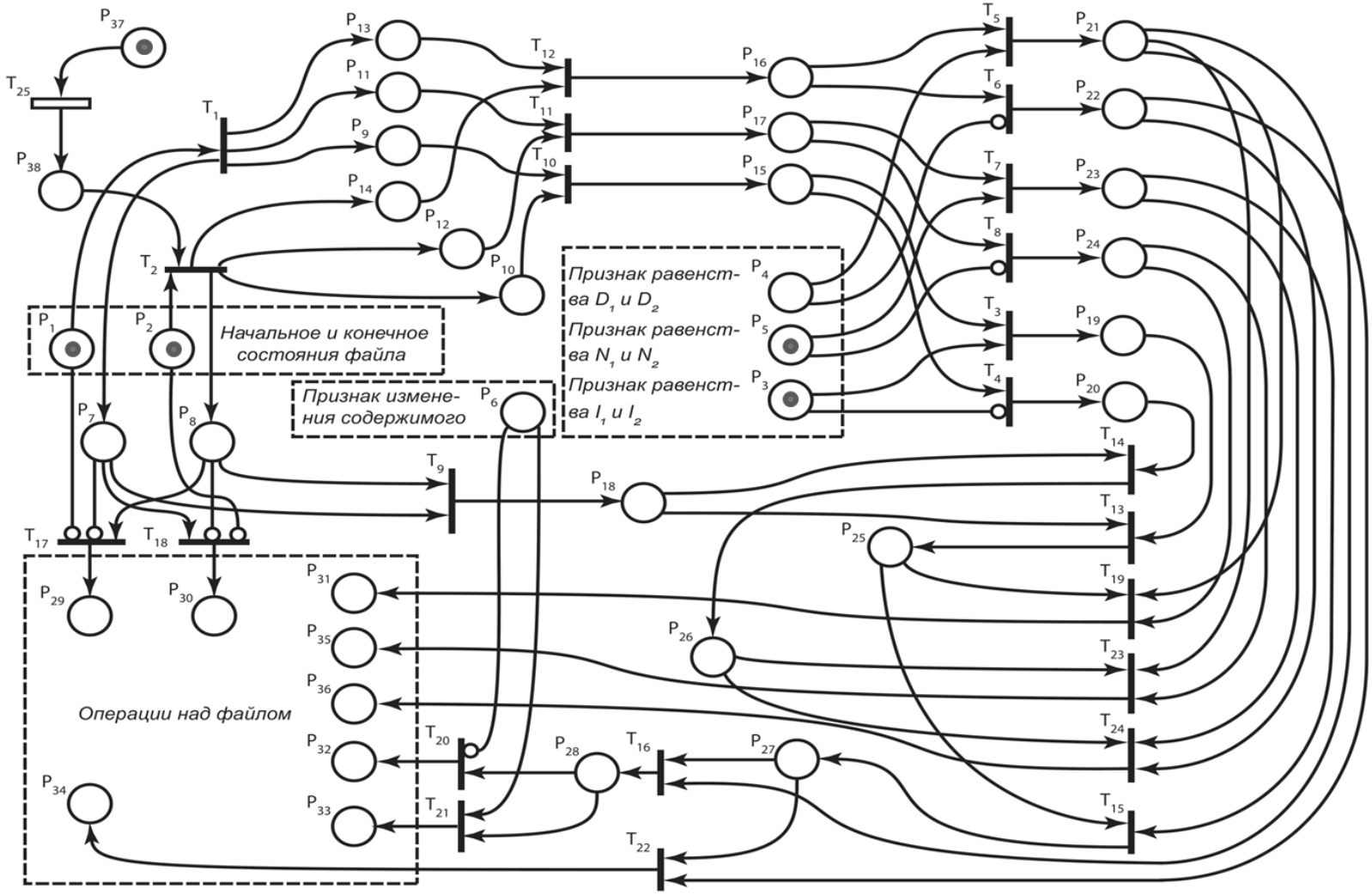


Рисунок Д.1. Задание начальной маркировки сети

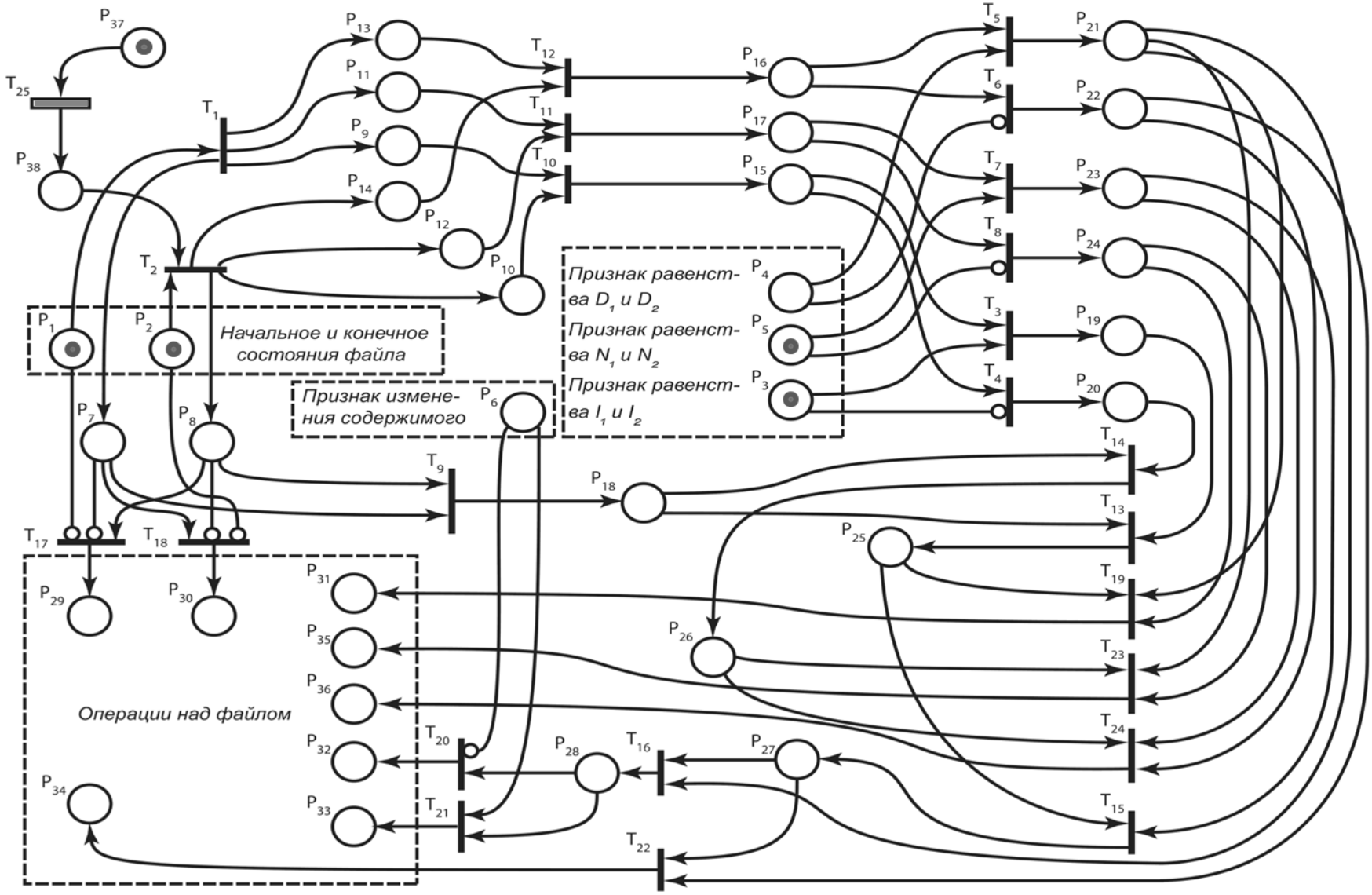


Рисунок Д.2. Пример активации перехода сети

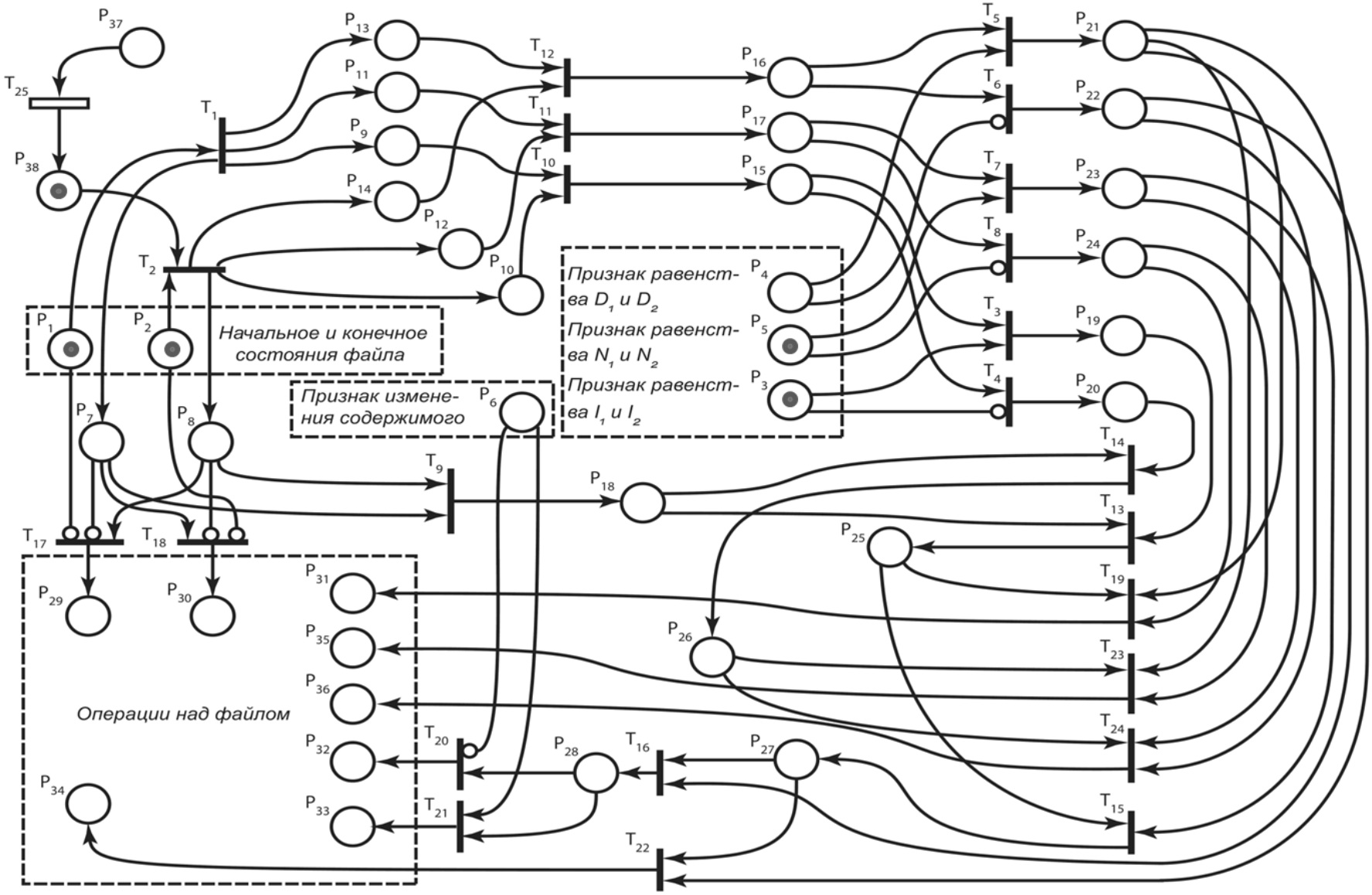


Рисунок Д.3. Позиции задания признаков, характеризующих файл

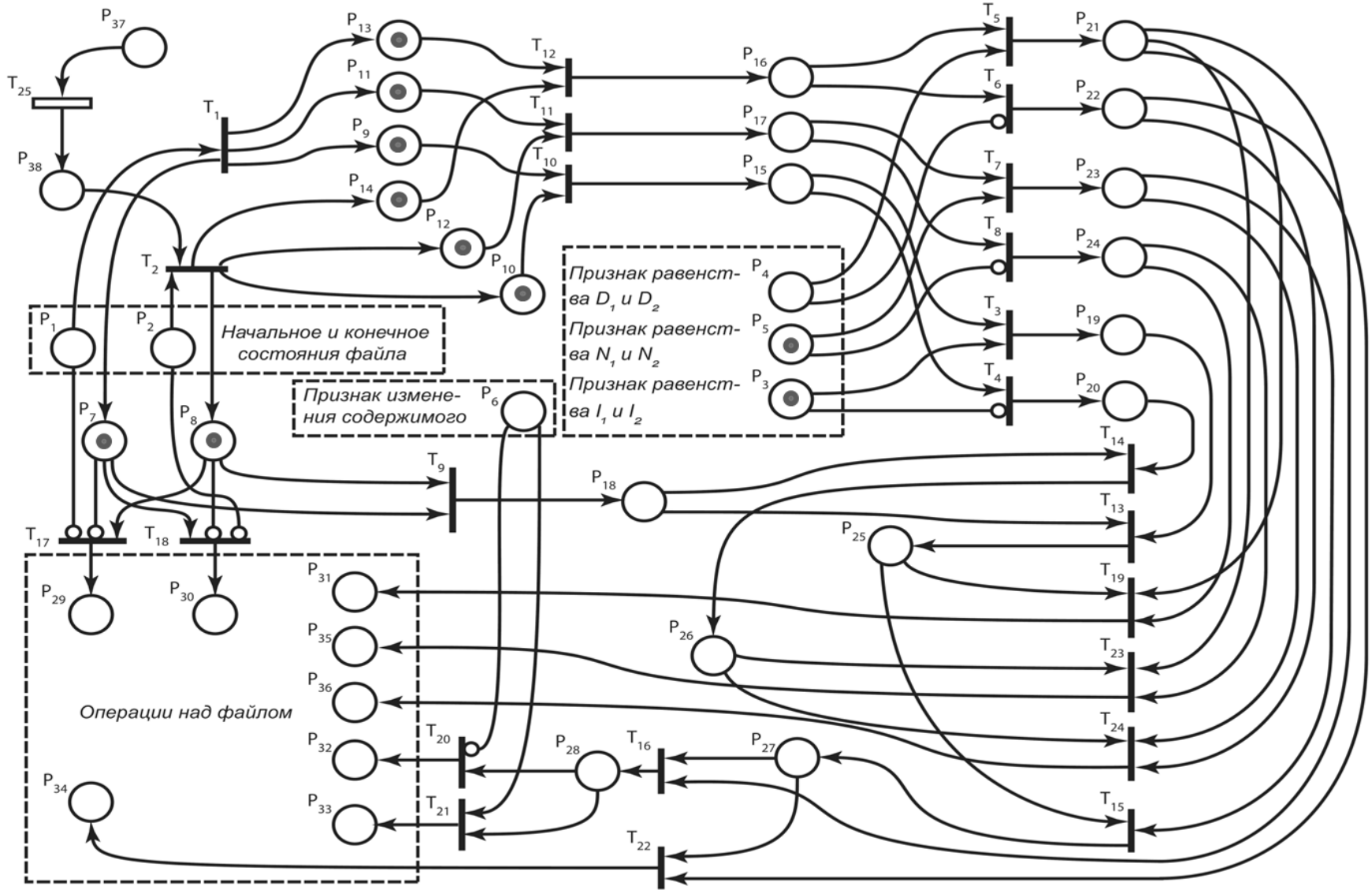


Рисунок Д.4. Исключение операций «Создание» и «Удаление»

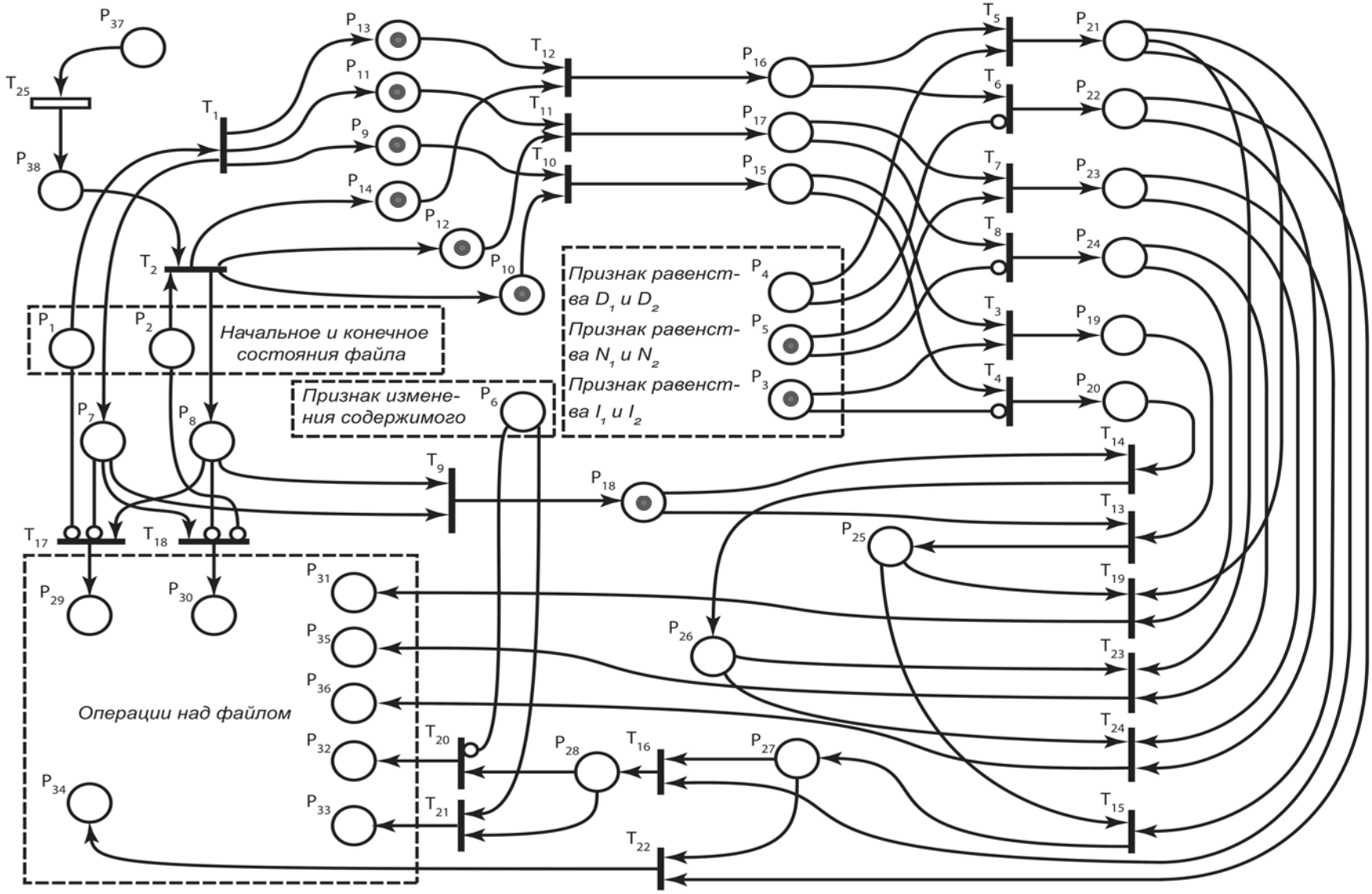


Рисунок Д.5. Определение файловых операций, связанных с копированием, переименованием, перемещением файла

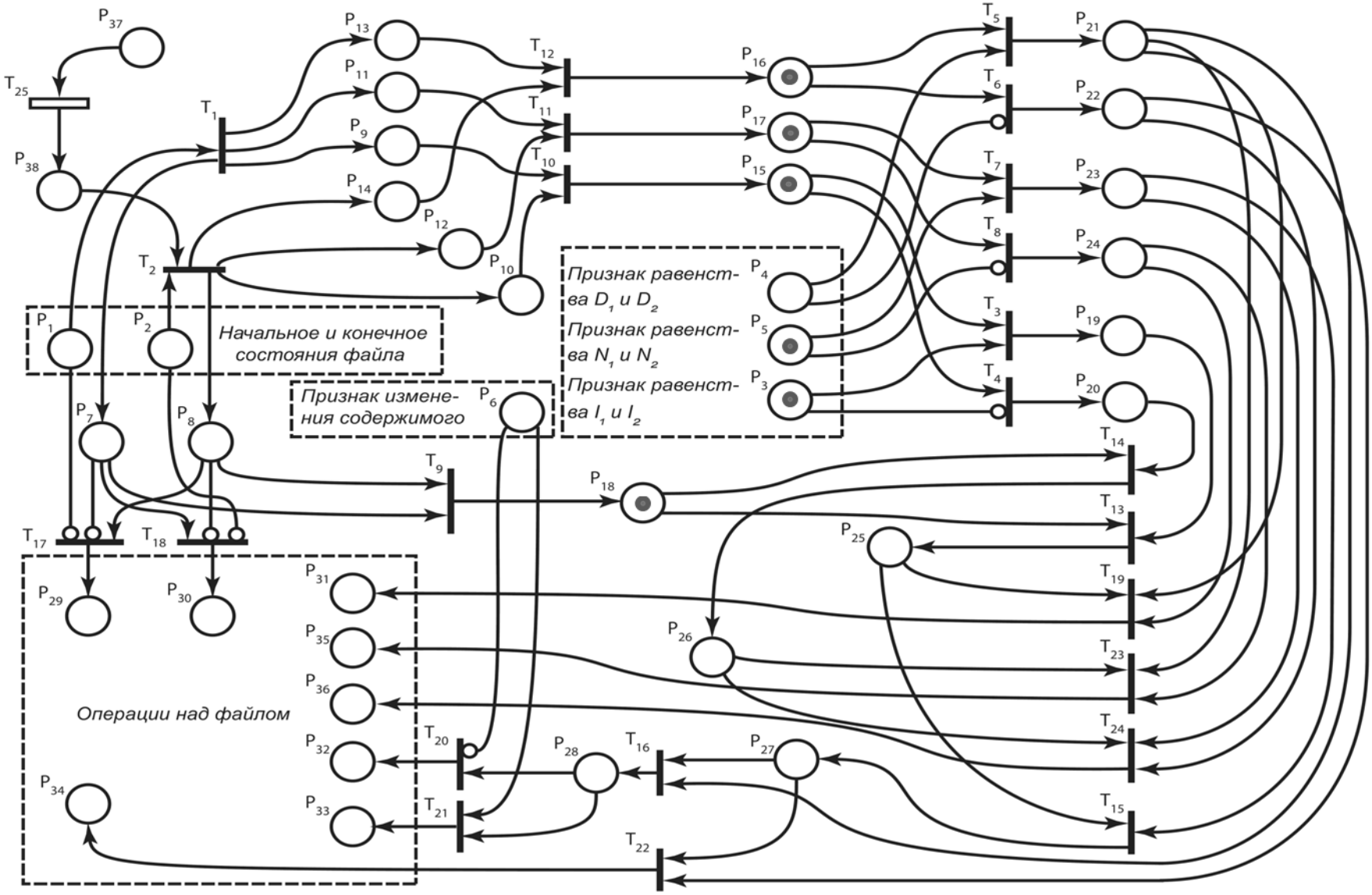


Рисунок Д.6. Исключение файловой операции, связанной с переименованием файла

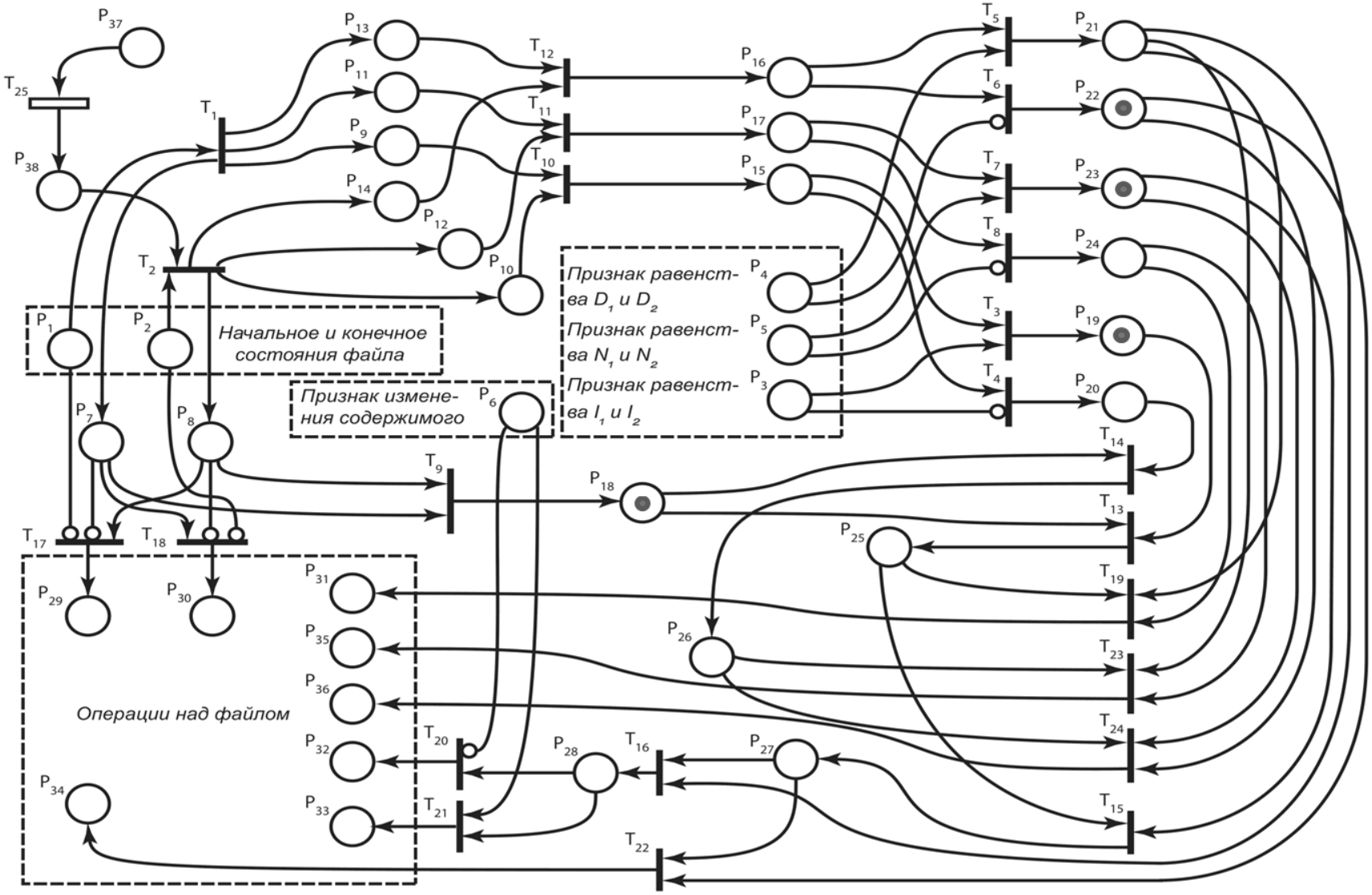


Рисунок Д.7. Исключение файловой операции, связанной с копированием файла в том же каталоге

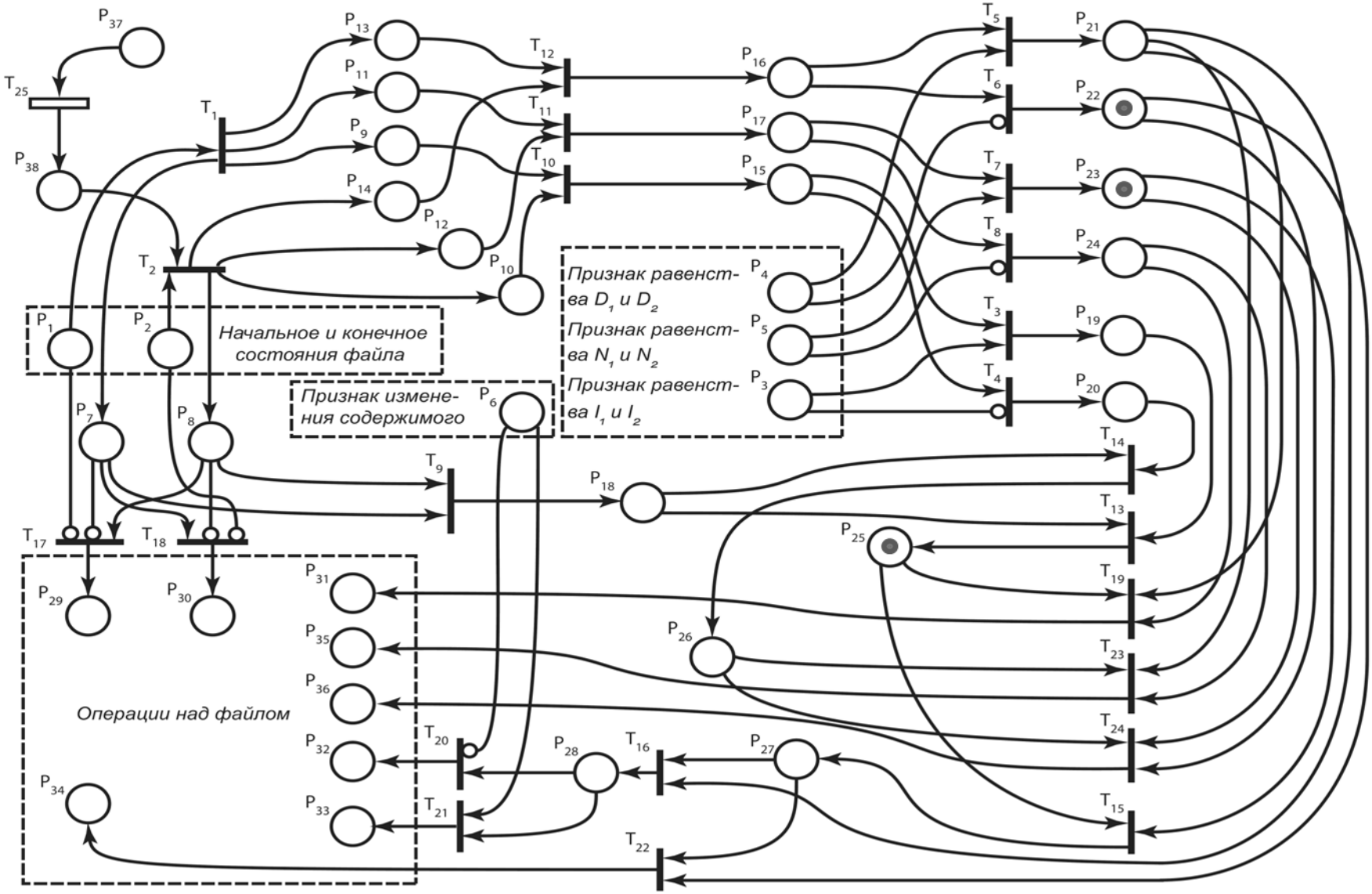


Рисунок Д.8. Исключение файловой операции, связанной с копированием файла в другой каталог

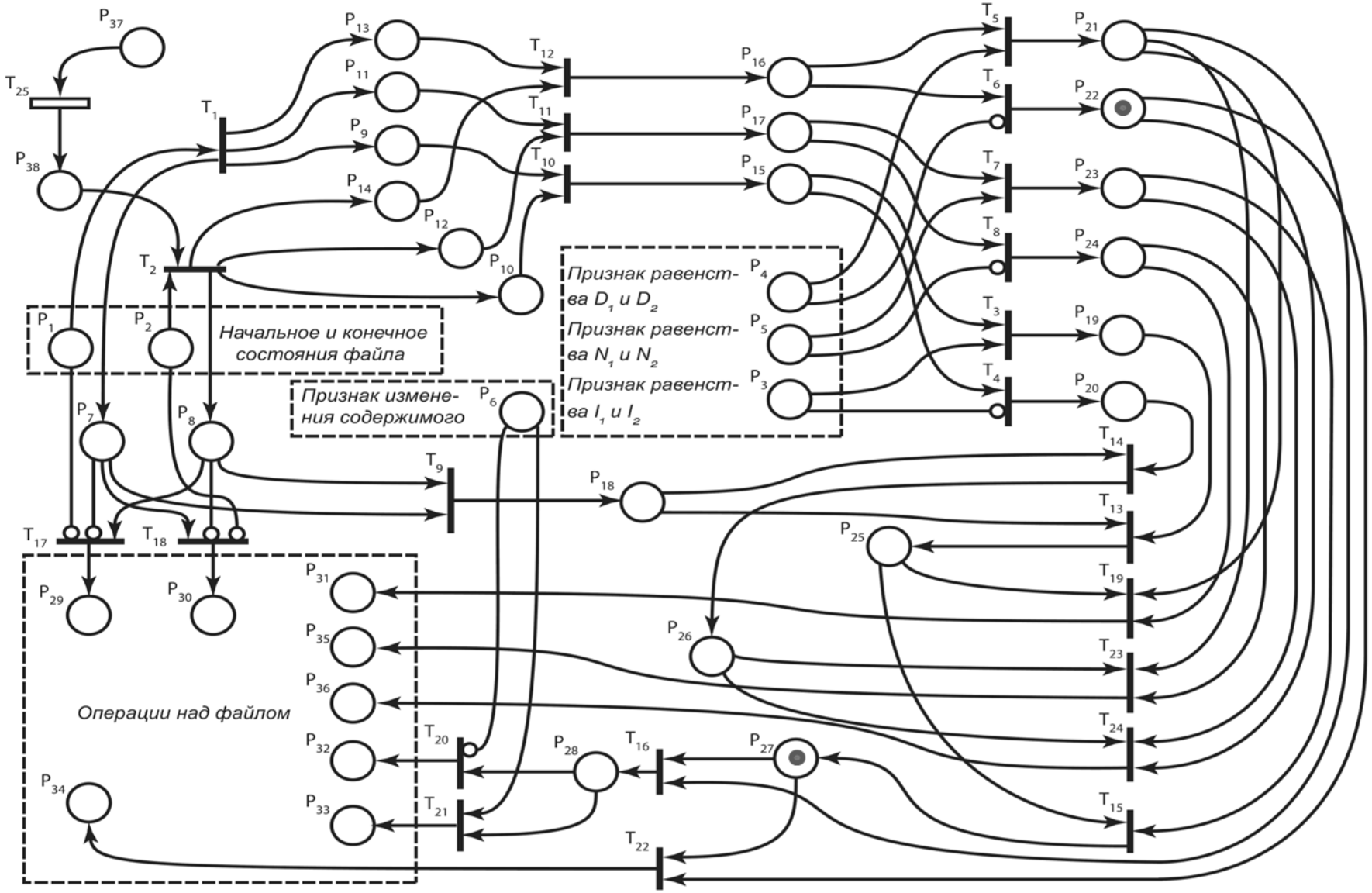


Рисунок Д.9. Исключение файловых операций, связанных с изменением содержимого и иной служебной информации

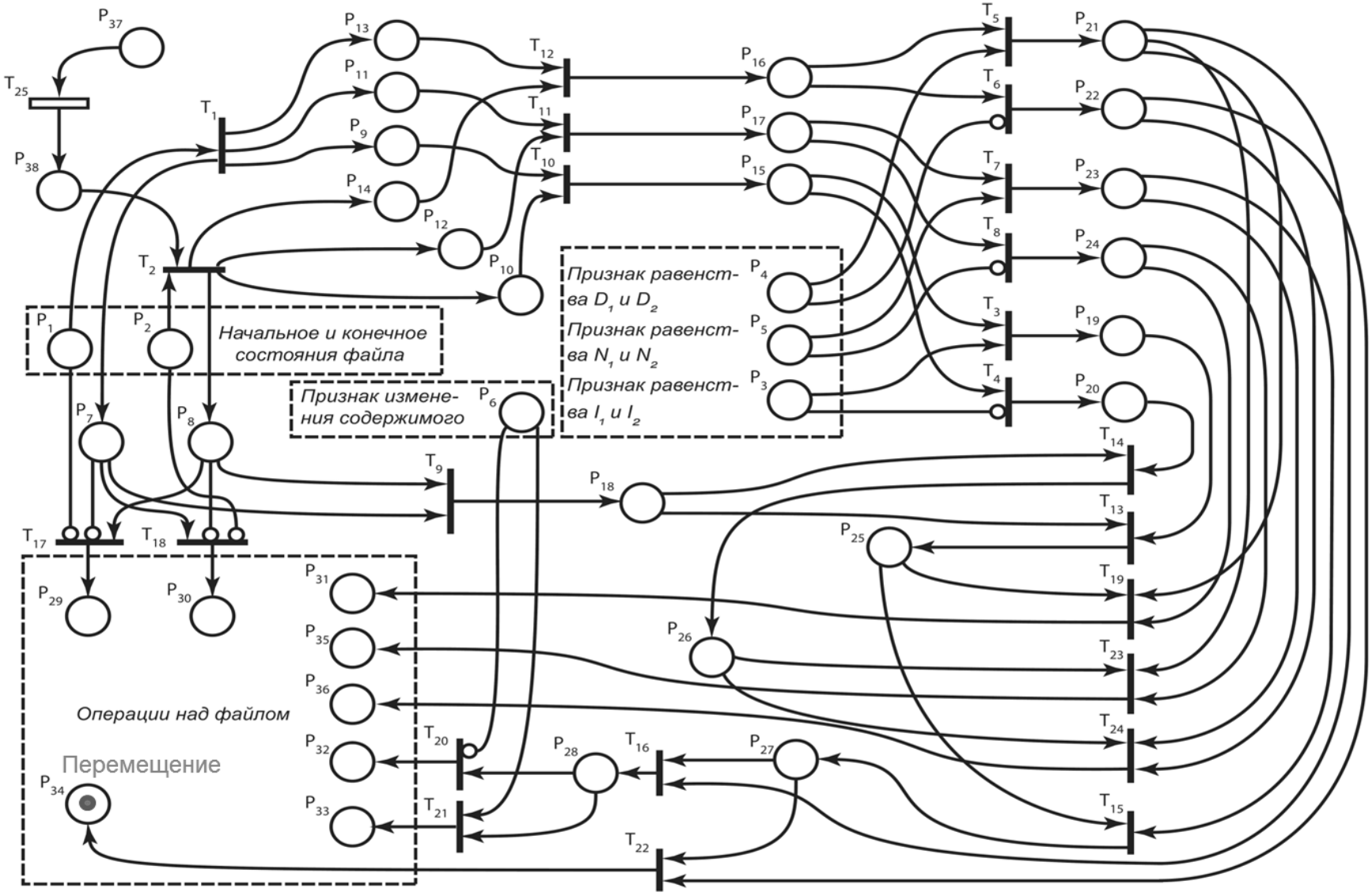


Рисунок Д.10. Классификация файловой операции «Перемещение»