УДК 004.056.57

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

В статье рассмотрены признаки, идентифицирующие файлы, приведены примеры файловых операций, предложен алгоритм идентификации файловых операций, представлена разработанная модель процесса идентификации воздействий на файлы. С учетом событийно-переходной природы процессов изменения признаков, идентифицирующих файлы, для моделирования использован аппарат сетей Петри. На основе модели возможна реализация системы, применение которой позволит автоматизировать процесс идентификации воздействий на файлы с целью определения влияния на обрабатываемую информацию в рамках расследования инцидентов информационной безопасности. Установление факта влияния на информацию позволяет упростить процедуру ликвидации последствий инцидента информационной безопасности. Использование системы на основе разработанной модели позволит верифицировать источники информации, содержащие признаки, идентифицирующие файлы, и данные о файловых операциях.

Ключевые слова: расследование инцидентов информационной безопасности, событие информационной безопасности, файловая операция, воздействие на файл, модель, сети Петри.

1. Введение

Информационные технологии (ИТ) оказывают влияние на множество областей развития современного общества: медицина, экономика, образование, промышленность, сфера услуг и др. Рассматривая положительные стороны влияния ИТ, не следует забывать о том, что их применение связано с рисками, а именно с возможностью совершения так называемых киберпреступлений. В целях минимизации рисков принимаются различные меры, направленные на обеспечение безопасности информации, обрабатываемой с применением ИТ обеспечения информационной безопасности (ИБ) и поиска новых средств и методов ее решения.

Одним из способов совершенствования мер, направленных на обеспечение ИБ, является расследование инцидентов ИБ. Результаты расследования учитываются при формировании рекомендаций по повышению эффективности принятых мер защиты информации. В процессе расследования определяются причины возникновения инцидента, регистрируется активность процессов, идентифицируются воздействия на файлы и информацию, хранящуюся в них, и др.

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

Расследовании инцидента ИБ возникает необходимость в разделении множества осуществленных воздействий на файлы на санкционированные и несанкционированные для ускорения процедуры ликвидации последствий инцидента. Для того чтобы разделить воздействия, их необходимо идентифицировать и оценить. В рамках статьи под идентификацией воздействий на файлы будем понимать процесс, в результате которого определяется порядок изменения параметров, характеризующих файл. Под оценкой воздействия на файл будем понимать процесс сопоставления идентифицированного воздействия с возможностью его появления согласно действующей политике безопасности компьютерной системы. Рассмотрение процесса оценки воздействий выходит за рамки статьи.

При расследовании инцидента ИБ в первую очередь интерес представляют несанкционированные воздействия на файлы. Специалист при выявлении несанкционированных воздействий может дать ответы на вопросы:

- нарушена ли конфиденциальность, целостность и/или доступность информации, хранящейся в файлах;
- какова активность пользователя и/или процесса по отношению к файлам;
- каков перечень файлов, подвергшихся несанкционированному воздействию.

Расследование инцидентов ИБ связано с анализом множества разноформатных массивов данных, присущих компьютерной системе и содержащих информацию о воздействиях на файлы (далее – массивы данных): временные отметки файлов [2], журналы событий операционной системы [3], журналы аудита [4], журналы средств защиты информации, записи о последних открытых файлах и т.д. Анализ таких массивов является основой процесса идентификации воздействий на файлы. В то же время использование указанных массивов связано с наличием ряда проблем:

- данные, содержащиеся в массиве, могут быть умышленно искажены. Этот факт свидетельствует о необходимости определения наличия искажений – требуется решение задачи верификации данных, содержащихся в массиве;
- данные, содержащиеся в различных массивах, не всегда содержат полный набор параметров, характеризующих файл. Отсутствие единого набора параметров не позволяет однозначно идентифицировать воздействие на файл – для решения задачи идентификации требуется формализация минимально необходимого набора параметров, которые могут быть получены из существующих массивов данных;
- в некоторых массивах отсутствуют данные, позволяющие идентифицировать сложные, комплексные воздействия на файлы.

Авторами в статье представлена разработанная событийная модель процесса идентификации воздействий на файлы, используемая как средство формализации данных, полученных из массивов при расследовании инцидентов ИБ. Предложенная модель позволяет идентифицировать воздействия на файлы, а также верифицировать данные, получаемые из массивов.

Понятийный аппарат, используемый при идентификации воздействий на файлы

В рамках настоящего исследования используется понятийный аппарат, детализирующий существующие определения инцидента ИБ и события ИБ в контексте решаемой задачи.

Причиной возникновения инцидента ИБ является событие ИБ, которое согласно [5] определяется как «идентифицированное появление определенного состояния системы, сервиса или сети, указывающего на возможное нарушение политики ИБ или отказ защитных

[1] В рамках статьи под верификацией массива данных будем понимать процесс выявления комбинаций параметров, характеризующих файл, возникновение которых невозможно в процессе штатного заполнения массива данными.

мер, или возникновение неизвестной ранее ситуации, которая может иметь отношение к безопасности». В соответствии с [1] инцидент ИБ определяется как совокупность событий ИБ. Опишем инцидент ИБ R выражением:

$$R = \langle S_1, \ldots, S_p \rangle,$$

где $S_1, \ldots, S_p$ – события ИБ, а p – их количество.

В рамках исследования каждое событие ИБ рассматривается с позиции идентификации воздействий на файлы и описывается кортежем:

$$S_k = \left\langle \left\langle A_{ji}^k \right\rangle_{i=1}^{l} \right\rangle_{j=1}^{n},$$

где A_{ji}^k – воздействие i на файл j, относящееся к событию ИБ k, l – количество воздействий, n – количество файлов.

Событие ИБ может сопровождаться широким спектром как простых, так и сложных комплексных воздействий на файлы. В основе каждого воздействия на файл лежат признаки, идентифицирующие файл, и соответствующая файловая операция. Таким образом, расследование инцидентов ИБ является процессом идентификации воздействий на файлы, составляющих событие ИБ.

Признаки, идентифицирующие файл

В рамках настоящего исследования будем считать, что произвольный файл j в каждый момент времени t идентифицируется следующими признаками:

- $I_j(t)$ – идентификатор файла – уникальное числовое значение, содержащееся в служебной информации, используемой драйвером файловой системы для однозначного определения файла;
- $D_j(t)$ – идентификатор родительского каталога файла – уникальное числовое значение, используемое драйвером файловой системы для установления однозначного соответствия между файлом и каталогом, в котором файл расположен;
- $N_j(t)$ – имя файла – битовая строка, используемая драйвером файловой системы для представления файла пользователю;
- $C_j(t)$ – содержимое файла – битовая строка, образующая информацию, хранимую в файле;
- $X_j(t)$ – иная служебная информация о файле – набор числовых значений, являющихся служебной информацией о файле, зависящий от типа файловой системы.

Признаки файла – битовая строка, образующая вектор $V_j(t)$, хранимый в файле, выражением:

$$V_j(t) = \{I_j(t), D_j(t), N_j(t), C_j(t), X_j(t)\}$$

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

$$V_j(t) = \{I_j, D_j, N_j, C_j, X_j\}$$

При рассмотрении некоторого файла j в рамках расследования инцидента ИБ возможно определение значений компонентов вектора идентифицирующих признаков V_j в текущий момент времени. Зачастую интерес представляет информация, описывающая изменения значений компонентов вектора идентифицирующих признаков V_j в ходе инцидента ИБ. Получить такую информацию можно при анализе указанных ранее массивов данных. Данные, появляющиеся в массивах в результате некоторого события, имеющего отношение к файлу, описывают состояние файла на момент события. Зададим определение состояния файла.

Определение 1. *Состояние файла – значения компонентов вектора идентифицирующих признаков V_j, присущих файлу в определенный момент времени t.*

2.2. Файловые операции

Изменение значений компонентов вектора идентифицирующих признаков V_j происходит не спонтанно, а в соответствии с процессом, называемым файловой операцией.

Пусть в момент времени t_1 начат процесс, в результате которого произошло изменение значений компонентов вектора V_j, характеризующего файл, и к моменту времени окончания

Зададим определение файловой операции.

Определение 2. *Файловая операция O – процесс модификации значений компонентов вектора идентифицирующих признаков V_j, приводящий к переходу файла от одного состояния к другому.*

Авторами предлагается рассмотреть один из вариантов зависимостей между файловыми

Таблица 1. Зависимости между файловыми операциями и изменяющимися компонентами V_j

Файловая операция	Изменяющиеся компоненты вектора V_j
Создание файла ($O1$)	I_j, D_j, N_j, X_j
Удаление файла ($O2$)	I_j, D_j, N_j, X_j
Переименование файла ($O3$)	N_j
Изменение содержимого файла ($O4$)	C_j, X_j[2]
Изменение служебной информации[3] ($O5$)	X_j
Перемещение файла в пределах логического раздела ($O6$)	D_j
Копирование файла в пределах каталога ($O7$)	1) I_j, D_j, N_j, X_j (создание конечного файла)[4]; 2) C_j, X_j (изменение содержимого конечного файла, изменение служебной информации конечного файла); 3) X_j (изменение служебной информации исходного файла).
Копирование файла в другой каталог (O8)	1) I_j, D_j, N_j, X_j (создание конечного файла)[5]; 2) C_j, X_j (изменение содержимого конечного файла, изменение служебной информации конечного файла); 3) X_j (изменение служебной информации исходного файла).

Примечание: исследование зависимостей проводилось в операционных системах Windows с файловой системой NTFS.

---

[2] Изменение размера файла повлечет за собой изменение служебной информации о файле. В случае, когда размер не изменяется, о факте изменения содержимого свидетельствует соответствующая временная отметка.

[3] Чтение файлов может сопровождаться изменением временной отметки последнего доступа к файлу при установке значений соответствующих параметров операционной системы, поэтому файловая операция «Чтение файла» относится к файловой операции «Изменение служебной информации». Исследование проводилось при установленном по умолчанию параметре операционной системы, связанном с запретом обновления временной отметки последнего доступа. В таком случае чтение файла не вызывает изменения признаков, идентифицирующих файл.

[4] У исходного и конечного файла совпадут D_j, но будут отличаться N_j.

[5] У исходного и конечного файла совпадут N_j, но будут отличаться D_j.

$V_j = V_j(t_1)$

К моменту начала расследования инцидента ИБ файловые операции считаются выполненными. Информация о них может быть получена из массивов данных. В связи с тем, что файловая операция в массивах не всегда указывается в явном виде, необходимо провести

$$O(t_1,t_2)=\left\langle V_j(t_1),V_j(t_2)\right\rangle$$

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

$$O=\left\langle V_{j1},V_{j2}\right\rangle$$

где V_{j1} и V_{j2} – векторы идентифицирующих признаков файла j для начального и конечного состояния соответственно.

При совершении множества файловых операций по отношению к файлу j происходит несколько переходов между состояниями $\{V_{j1},V_{j2},...,V_{jm*},...,V_{jm}\}$, причем при каждом переходе состояние $m*$ считается конечным, а $m*-1$ – начальным.

Исходя из определений состояния файла и файловой операции, переходы между

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. В рамках исследования изучены особенности некоторых типов воздействий на файлы, совершение которых привело 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.

Определение воздействия на файл с учетом выражения (6) описывается как:

где $O_{jm'}$ – файловая операция, осуществляемая в отношении файла j, $V_{jm'}$ – вектор признаков,

$$A_i=\left\langle\left\langle O_{jm'}\right\rangle_{m'=1}^{m-1}\right\rangle_{j=1}=\left\langle\left\langle V_{jm'},V_{j(m'+1)}\right\rangle_{m'=1}^{m-1}\right\rangle_{j=1},$$

Воздействие на файл может представлять собой одну операцию (j = 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 комплексным ($j \geq 1$, $m' > 1$). Примерами простых воздействий на файлы являются:

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

Примеры комплексных воздействий на файлы:

- изменение содержимого документа формата doc, docx, odt в текстовых процессорах

- 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

Алгоритм идентификации файловой операции

Алгоритм может быть использован как для идентификации одной файловой операции (при отсутствии в массиве данных информации о файловой операции), так и для верификации данных массива (в целях обеспечения достоверности). Необходимость в идентификации/верификации последовательности файловых операций удовлетворяется представленной далее моделью. Входными данными алгоритма являются значения компонентов векторов $V_{j1} = \{ I_{j1}, D_{j1}, N_{j1}, C_{j1}, X_{j1} \}$ и $V_{j2} = \{ I_{j2}, D_{j2}, N_{j2}, C_{j2}, X_{j2} \}$, характеризующих начальное и конечное состояния файла j соответственно.

Алгоритм состоит из нескольких шагов:

1. П
2. Определить принадлежность I_{j1} и I_{j2} пустому множеству.
3. Е
4. Если $I_{j1} \notin \varnothing$ и $I_{j2} \in \varnothing$, то идентифицируется операция «Удаление файла» (табл. 1, $I_{j1} \varnothing$.
5. Если $I_{j1} \notin \varnothing$ и $I_{j2} \notin \varnothing$, то необходимо провести дальнейшее сравнение признаков, идентифицирующих файл.
6. При несовпадении I_{j1} и I_{j2} определяется одна из двух операций копирования (табл. 1, $I_{j2} \varnothing$, то идентифицируется операция «Создание файла» (табл. 1, O1).
7. В
8. Если не равны имена файла, но равны идентификаторы его родительского каталога, то идентифицируется операция «Переименование файла» (табл. 1, O3).
9. Если равны имена файла, но не равны идентификаторы его родительского каталога, то идентифицируется операция «Перемещение файла в пределах логического раздела» (табл. 1,
10. В случае совпадения имен файла и идентификаторов его родительского каталога
11. Е

Результатом работы алгоритма является идентификация файловой операции, совершенной над файлом. Стоит отметить, что элемент «Вызов исключения» является условным и указывает

Блок-схема алгоритма идентификации файловой операции представлена на рис. 1.

При применении алгоритма следует учитывать ряд особенностей:

1. $\neg\exists V_j \mid I_j \notin \varnothing, D_j \in \varnothing$ (файл j всегда расположен в его родительском каталоге).

равны, то идентифицируется операция «Изменение служебной информации» (табл. 1, O5), в противном случае – «Изменение содержимого файла» (табл. 1, O4).

3. Из рис. 1 видно, что для идентификации всех файловых операций отсутствует необходимость сравнения значений компонентов X_{j1} и X_{j2} ввиду отсутствия неопре-

деленности в работе алгоритма. В связи с этим, при разработке модели компоненты X_j и C_j заменены на признак EMBEDEquation $Z_j = f(X_j, C_j)$ – 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_j принимает два значения – 1, если изменилось содержимое файла, и 0, если изменилась его служебная информация.

Собственная модель процесса идентификации вредоносной активности на файлы

…

PAGEMER

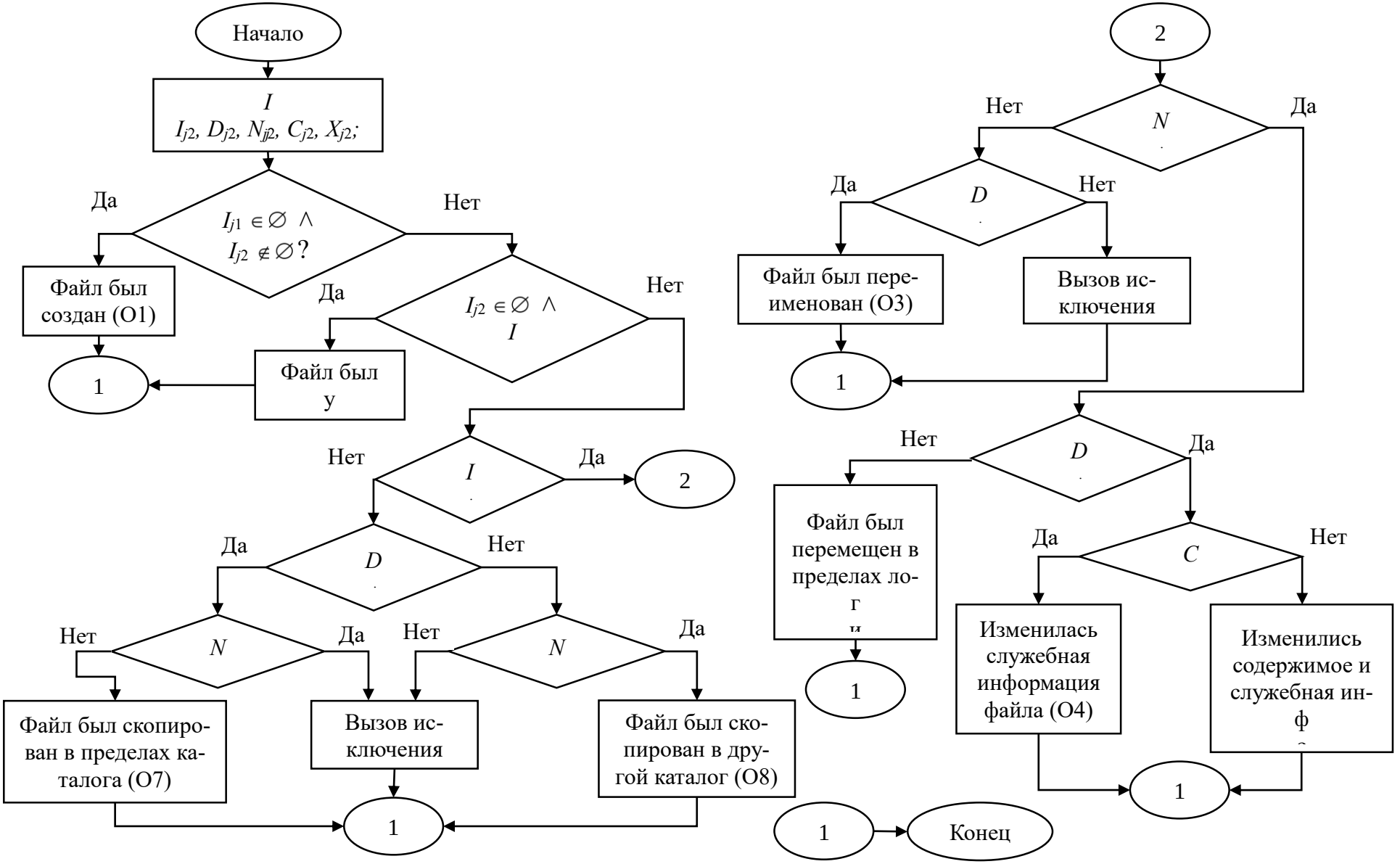

Рис. 1. Блок-схема алгоритма идентификации файловой операции

Модель процесса идентификации воздействий на файлы

Представленный алгоритм предназначен для идентификации файловой операции и 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.

Для построения модели с использованием выражения (5) и алгоритма, изображенного на рис. 1, в рамках введенных определений воспользуемся математическим аппаратом сетей Петри, хорошо зарекомендовавшим себя в качестве инструмента формализации алгоритмов и процессов [6, 7], который позволит смоделировать процесс идентификации воздействий на файлы. Смоделированная сеть построена таким образом, чтобы исключить случайные переходы.

Сеть Петри задается пятью параметрами – *P, T, E, Q, μ*. Представленная модель процесса идентификации воздействий на файлы *M,* построенная с использованием сети Петри, дополняется двумя параметрами: *δ*, который является внешним по отношению к сети «накопителем» файловых операций, и временной задержкой $\tau = t_2 - t_1$, где t_1 соответствует

1. В конечное множество позиций $P = P_{fs} \cup P_{al} \cup P_{fin} \cup P_{add}$ включены:
   - п
   - позиции состояний алгоритма P_{al} ($P_3 - P_5$, $P_{19} - P_{28}$), представленного на рис. 1
   - позиции «индикации» идентифицированных 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);
   - п
   - вспомогательные переходы, предназначенные для корректного функционирования сети Петри T_{add} ($T_9 - T_{12}$);
   - п
   - переходы идентификации файловых операций T_{fo}

М

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 со вспомогательными позициями и переходами.

4. М

5. Конечное множество значений временных задержек нетривиальных переходов τ

Начальное состояние сети Петри, обозначаемое как *μ,* 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

$T_{17} \in O1, T_{18} \in O2, T_{19} \in O3, T_{20} \in O5, T_{21} \in O4, T_{22} \in O6, T_{23} \in O7, T_{24} \in O8$

(типы файловых операций представлены в табл. 1).

по отношению к моделируемому процессу элементом. Позиции $P_7 – P_{28}$ и переходы $T_3 – T_{24}$ реализуют ветвления алгоритма идентификации файловой операции, представленного на рис. 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» δ, 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 (7). 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 файловой операции с другим файлом в рамках одного сложного комплексного воздействия. Конечное множество маркировок, обеспечивающих определение файловой операции, представлено в табл. 2.

Таблица 2. Множество маркировок позиций

Маркировка μ	Файловая операция
P	Создание файла (табл. 1, $O1$)
P	Удаление файла (табл. 1, $O2$)
P	Переименование файла (табл. 1, $O3$)
P	Изменение содержимого файла (табл. 1, $O4$)
P	Изменение служебной информации файла (табл. 1, $O5$)
P	Перемещение файла в пределах логического раздела (табл. 1, $O6$)
P	Копирование файла в пределах каталога (табл. 1, $O7$)
P	Копирование файла в другой каталог (табл. 1, $O8$)

Разработанная модель изображена на рис. 2. Описание позиций и переходов представлено в табл. 3 и 4 приложения соответственно.

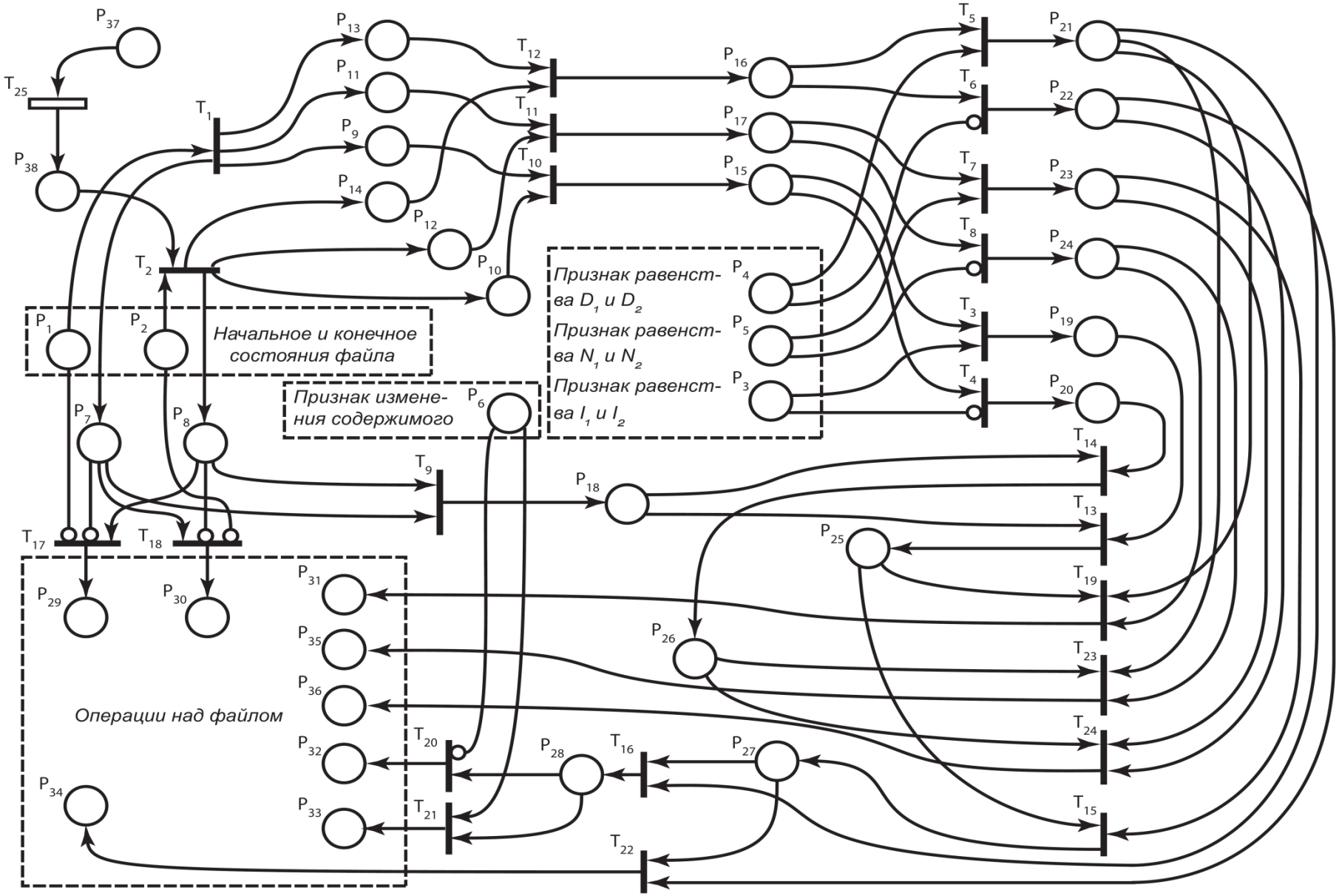

Рис. 2. Сеть Петри, описывающая модель процесса идентификации воздействий на файлы

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

1. Операции «Создание файла» и «Удаление файла» определяются независимо от

2. Операции «Изменение служебной информации файла» и «Изменение содержимого

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 табл. 1. Они представлены отдельными позициями сети, не зависящими от операций «Создание файла» и «Изменение содержимого файла».

Представленная сеть смоделирована и протестирована в специализированном программном обеспечении Platform Independent Petri Net Editor [8, 9]. Соответствие результата, полученного после выполнения сети, реальным файловым операциям экспериментально подтверждено на специальном стенде с установленными операционными системами Microsoft

Заключение

Использование разработанной модели позволяет реализовать систему идентификации воздействий на файлы, применение которой возможно при расследовании инцидентов ИБ. Получив информацию о начальном и конечном состояниях файла, специалист, осуществляющий расследование инцидента ИБ, задает начальное состояние маркировки сети фишками и по результатам однократного или множественного выполнения сети идентифицирует осуществленные воздействия на файлы.

Практическое применение разработанной модели рассматривается с позиции решения двух задач – идентификации и верификации. При необходимости идентификации воздействий на файлы модель может быть использована в качестве основы для создания автоматизированной системы контроля состояний файлов с учетом большого количества данных в массивах. В случае необходимости верификации данных, полученных из массивов, модель может быть применена как средство контроля за целостностью этих данных.

Приложение

1. Описание позиций представленной сети Петри

Таблица 3. Описание позиций

№ позиции	Описание
Позиции маркировки, определяющие начальное состояние сети Петри	
P_1	Д
P_2	Д
P_3	Условие равенства I_{j1} и I_{j2}. Установка фишки, если $I_{j1} = I_{j2}$
P_4	Условие равенства D_{j1} и D_{j2}. Установка фишки, если $D_{j1} = D_{j2}$
P_5	Условие равенства N_{j1} и N_{j2}. Установка фишки, если $N_{j1} = N_{j2}$
P_6	Признак изменения содержимого файла. Установка фишки необходима, если содержимое файла изменялось

№ позиции	Описание
P_{37}	Имитация задержки осуществления файловой операции. Установка фишки необходима для запуска перехода T_{25}
Вспомогательные и алгоритмические позиции	
P	Файл 1 существует
P	Файл 2 существует
P_9	$I_{j1} \notin \varnothing$
P_{10}	$I_{j2} \notin \varnothing$
P_{11}	$D_{j1} \notin \varnothing$
P_{12}	$D_{j2} \notin \varnothing$
P_{13}	$N_{j1} \notin \varnothing$
P_{14}	$N_{j2} \notin \varnothing$
P_{15}	$I_{j1}, I_{j2} \notin \varnothing$
P_{16}	$D_{j1}, D_{j2} \notin \varnothing$
P_{17}	$N_{j1}, N_{j2} \notin \varnothing$
P_{18}	Файлы 1 и 2 существуют
P_{19}	$I_{j1} = I_{j2}$
P_{20}	$I_{j1} \neq I_{j2}$
P_{21}	$D_{j1} = D_{j2}$
P_{22}	$D_{j1} \neq D_{j2}$
P_{23}	$N_{j1} = N_{j2}$
P_{24}	$N_{j1} \neq N_{j2}$
P_{25}	$I_{j1}, I_{j2} \notin \varnothing$ и $I_{j1} = I_{j2}$
P_{26}	$I_{j1}, I_{j2} \notin \varnothing$ и $I_{j1} \neq I_{j2}$
P_{27}	$I_{j1}, I_{j2} \notin \varnothing$, $I_{j1} \neq I_{j2}$, $N_{j1} \neq N_{j2}$
P	$I_{j1}, I_{j2} \notin \varnothing$, $I_{j1} \neq I_{j2}$, $N_{j1} = N_{j2}$, $D_{j1} = D_{j2}$
P	Имитация завершения файловой операции, после задержки в T_{25}
Позиции маркировки, указывающие на идентифицированную файловую операцию	
P	Создание файла (табл. 1, $O1$)
P	Удаление файла (табл. 1, $O2$)
P	Переименование файла (табл. 1, $O3$)
P_{32}	Изменение служебной информации файла (табл. 1, $O5$)
P_{33}	Изменение содержимого файла (табл. 1, $O4$)
P_{34}	Перемещение файла в пределах логического раздела (табл. 1, $O6$)
P	Копирование файла в пределах каталога (табл. 1, $O7$)
P	Копирование файла в другой каталог (табл. 1, $O8$)

Описание переходов представленной сети Петри

Таблица 4. Описание переходов

№ перехода	Описание
T	П
T	П
Вспомогательные и алгоритмические переходы	
T_3	П
T_4	Получение значения – неравенство I_{j1} и I_{j2}
T_5	Получение значения – равенство D_{j1} и D_{j2}
T_6	Получение значения – неравенство D_{j1} и D_{j2}
T_7	Получение значения – равенство N_{j1} и N_{j2}
T_8	П
T_9	Исключение операций «Создание файла» и «Удаление файла»
T_{10}	О
T_{11}	Определение принадлежности D_{j1} и D_{j2} к непустому множеству
T_{12}	Определение принадлежности N_{j1} и N_{j2} к непустому множеству
T_{13}	Переход к файловым операциям с равными признаками I_{j1} и I_{j2}
T_{14}	Переход к файловым операциям с неравными признаками I_{j1} и I_{j2}
T_{15}	П
T	П
T	Непримитивный переход, имитирующий время выполнения файловой операции
Переходы, идентифицирующие файловые операции	
T	Создание файла (табл. 1, $O1$)
T	Удаление файла (табл. 1, $O2$)
T	Переименование файла (табл. 1, $O3$)
T_{20}	Изменение служебной информации файла (табл. 1, $O5$)
T_{21}	Изменение содержимого файла (табл. 1, $O4$)
T_{22}	Перемещение файла в пределах одного логического раздела (табл. 1, $O6$)
T	Копирование файла в рамках каталога (табл. 1, $O7$)
T	Копирование файла в другой каталог (табл. 1, $O8$)

Литература

1. 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
2. *Бакланов В. В., Князева Н. С., Хорьков Д. А.* Анализ временных отметок файловой системы NTFS в операционной системе Microsoft Windows XP // Проблемы информационной безопасности. Компьютерные системы. 2012. № 4. С. 25–32.
3. *Dwyer J., Marius Truta T.* Finding Anomalies in Windows Event Logs Using Standard Deviation // 9th IEEE International on Collaborative Computing: Networking, Applications and Worksharing, 2013. P. 563–570.
4. *Studiawan H., Payne C., Sohel F.* Graph Clustering and Anomaly Detection of Access Control Log for Forensic Purposes // Digital Investigation. 2017.
5. ГОСТ Р ИСО/МЭК ТО 18044-2007. Информационная технология. Методы и средства обеспечения безопасности. Менеджмент инцидентов информационной безопасности. М.: ФГУП «Стандартинформ», 2007. 50 с.
6. *Effendi Y. A., Sarno R.* Discovering Process Model from Event Logs by Considering Overlapping Rules // EECSI 2017. Yogyakarta, Indonesia, 2017.
7. *Хорьков Д. А., Гайдамакин Н. А.* Модель атакующего воздействия на автоматизированные системы в рамках развития аппарата сетей Петри // Проблемы информационной безопасности. Компьютерные системы. 2013. № 1. С. 73–80.
8. GitHub – sarahtattersall/PIPE: PIPE – Platform Independent Petri Net Editor [Электронный ресурс]. URL: https://github.com/sarahtattersall/PIPE (дата обращения:
9. *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. P. 34–39.

Статья поступила в редакцию 30.01.2020.

Гайдамакин Николай Александрович
д.т.н., профессор, профессор учебно-научного центра «Информационная безопасность» ИРИТ-РтФ УрФУ им. первого Президента России Б.Н. Ельцина (620002, Екатеринбург, ул. Мира, 19) e-mail: n.a.gaydamakin@urfu.ru.

Гибилинда Роман Владимирович
ассистент учебно-научного центра «Информационная безопасность» ИРИТ-РтФ УрФУ

Синадский Николай Игоревич
к.т.н., доцент, доцент учебно-научного центра «Информационная безопасность» ИРИТ-РтФ УрФУ им. первого Президента России Б.Н. Ельцина (620002, Екатеринбург, ул. Мира, 19) e-mail: n.i.sinadskiy@urfu.ru.

Event model of file impact identification process during information security incident response based on Petri Nets

N. A. Gaydamakin, R. V. Gibilinda, N. I. Sinadskiy

The article introduces file identifying characteristics, describes file operation examples, offers file operation identification algorithm, presents developed file impact identification process model. There is a possibility to create system based on developed model that allows to automate file impact identification process to determine impact on stored information during information security incident response. Information impact determination makes incident result removing process easier. The system used allows to verify information sources that contain file identifying characteristics and file operations data.

Keywords: information security incident response, information security event, file operation, file impact, model, Petri nets.

Н. А. Гайдамакин, Р. В. Гибилинда, Н. И. Синадский