Модель процесса выявления следов использования программного обеспечения в операционных системах macOS при расследовании инцидентов информационной безопасности

Р. В. Гибилинда, Н. С. Князева, К. С. Семенова

***Аннотация* — В статье представлена модель процесса выявления следов использования программного обеспечения, используемая при расследовании инцидентов информационной безопасности в операционных системах macOS. Акцент делается на примерах выявления следов запуска вредоносного программного обеспечения. Входными данными предлагаемой модели являются массивы данных, содержащие интересующие специалиста признаки, связанные с запуском программы: имя (полный путь до) файла, время запуска и источник информации о запуске. В статье кратко описаны структуры рассматриваемых массивов данных, приведены примеры идентификации программного обеспечения по косвенным признакам, связанным с созданием и открытием файлов. Предлагаемая авторами модель может быть расширена с использованием дополнительных (в т. ч. новых) массивов данных, а также использована в качестве основы по созданию программного обеспечения автоматического анализа информации для ускорения процесса расследования инцидента информационной безопасности.**

***Ключевые слова* — расследование инцидентов информационной безопасности, программа, следы использования, IDEF0, macOS.**

I. Введение

Операционная система (ОС) macOS является второй по популярности в мире после Windows и занимает долю рынка в 14,68% [1]. Использование macOS связано как с личными пользовательскими предпочтениями, так и рабочей необходимостью, например, в IT-компаниях macOS необходима для полноценной разработки и отладки программного обеспечения (ПО) для iPhone и iPad.

Второе место по распространенности рассматриваемой ОС не дает покоя злоумышленникам, занимающимся разработкой вредоносного ПО. Пользователь, работая в macOS, может сформировать у себя ложное чувство «защищенности», предполагая, что может не использовать антивирусное ПО и пренебрегать компьютерной «гигиеной». Вместе с тем, в сети Интернет существуют различные классы вредоносного ПО [2], многие из которых не потеряли актуальности даже в последних версиях macOS. Заражение компьютера под управлением macOS «бэкдором» или вирусом-«шифровальщиком» может привести к инциденту информационной безопасности (ИБ), связанному с потерей конфиденциальности, целостности и/или доступности данных.

Для выявления следов использования ПО в рамках расследования инцидента ИБ на локальном компьютере необходимо проводить анализ [3] специфических разноформатных массивов данных [4]–[7], в т.ч. с использованием средств автоматизации извлечения и анализа данных. Вместе с тем, для уменьшения ошибок анализа, связанных с т.н. «человеческим» фактором, возникает потребность в формализации процесса определения следов использования ПО.

Авторами предлагается модель такого процесса на основании IDEF0-нотации. Для формирования модели необходимо классифицировать исследуемые массивы данных, изучить их формат. Следование процессу, описываемому предложенной моделью при расследовании инцидентов ИБ позволит сформировать ход инцидента в привязке ко времени.

II. Классификация массивов данных

На основании проведенного информационного поиска и поставленных экспериментов, проанализированные массивы данных были отнесены к двум классам:

– непосредственно содержащие информацию о запуске программ – класс 1;

– содержащие информацию об открытых файлах,

Статья получена 18 февраля 2025.

Роман Владимирович Гибилинда, кандидат технических наук, доцент учебно-научного центра «Информационная безопасность», Институт радиоэлектроники и информационных технологий-РтФ ФГАОУ ВО «УрФУ им. первого Президента России Б.Н. Ельцина» (e-mail: r.v.gibilinda@urfu.ru).

Наталия Сергеевна Князева, кандидат технических наук, доцент учебно-научного центра «Информационная безопасность», Институт радиоэлектроники и информационных технологий-РтФ ФГАОУ ВО «УрФУ им. первого Президента России Б.Н. Ельцина» (e-mail: n.s.kniazeva@urfu.ru).

Ксения Сергеевна Семенова, ассистент учебно-научного центра «Информационная безопасность», Институт радиоэлектроники и информационных технологий-РтФ ФГАОУ ВО «УрФУ им. первого Президента России Б.Н. Ельцина» (e-mail: xeniasemenova2009@yandex.ru).

среди которых могут встретиться программы – класс 2.

Предлагаемая авторами классификация массивов данных представлена на рисунке 1.

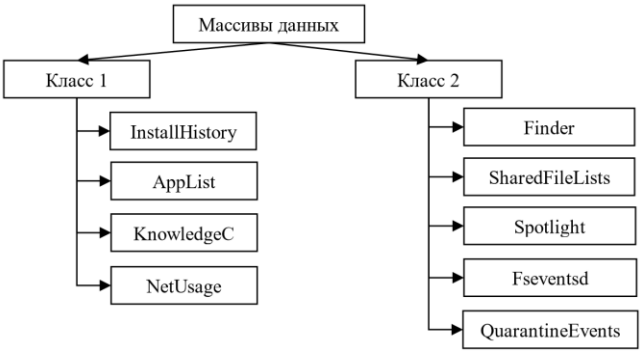

Рис. 1. Классификация массивов данных

III. Массивы данных, содержащие информацию о запускаемых приложениях

Файл InstallHistory.plist [8] имеет формат plist (формат plist основан на языке разметки XML, содержит два базовых типа сущностей «ключ» («key») и «значение» («value»). Ключи могут содержать в себе другие ключи, имеют имена и значения. Значения интерпретируются в зависимости от типа: строка («string»), массив («data»), словарь («dict»), логическое значение («boolean»), дата («date») и т.д.) и располагается по пути /Library/Receipts. Для каждой установленной программы в файле хранится ключ, содержащий ключи date (дата установки программы), displayName (название программы или обновления), displayVersion (версия программы), processName (имя процесса, инициирующего установку или обновление программы). Фрагмент содержимого файла InstallHistory.plist представлен на рисунке 2. Для автоматизированного извлечения информации можно воспользоваться программой mac_apt [9] с плагином «INSTALLHISTORY», а для просмотра содержимого файла InstallHistory.plist — программой plist Editor.

Следующий массив данных appList.dat [8] имеет формат bplist (формат хранения текстовых xml-данных в двоичном формате) и располагается по пути /Users/%username%/Library/Application Support/ com.apple.spotlight, где %username% — имя пользователя. Для каждой установленной программы (даже после ее удаления) в файле хранится ключ, содержащий ключи displayName (название программы) и URL (полный путь к исполняемому файлу). Для автоматизированного извлечения информации из файла appList.dat можно воспользоваться программой mac_apt с плагином APPLIST, а для просмотра содержимого, предварительно преобразовав в формат plist, — программой plist Editor.

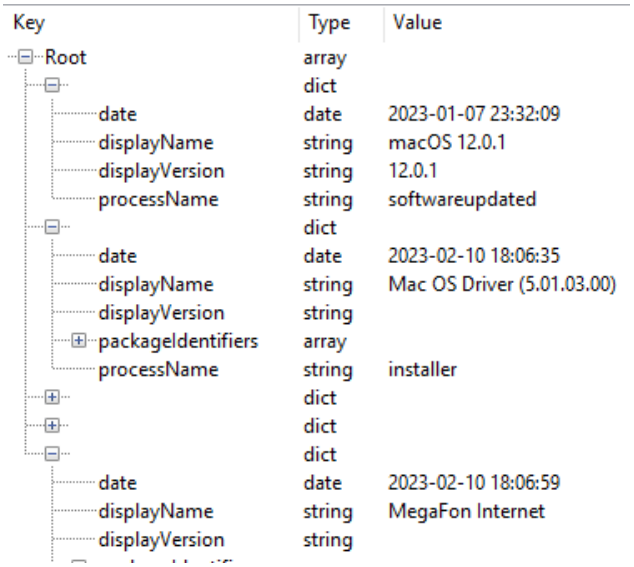

Рис. 2. Фрагмент содержимого файла InstallHistory.plist

Массив KnowledgeC.db [10] является базой данных формата sqlite и располагается по пути /private/var/db/CoreDuet/Knowledge и /Users/ %username%/Library/Application Support/ Knowledge. В базе данных хранятся записи об активности программ: наименование запущенной программы (поле ZBUNDLEID таблицы ZSOURCE), дата и время запуска (поле ZSTARTDATE таблицы ZOBJECT), дата и время окончания работы (поле ZENDDATE таблицы ZOBJECT), а также дополнительные сведения о работе программы, например, имя редактируемой заметки, URL посещаемой страницы в Web-браузере (поля ZSTREAMNAME и ZVALUESTRING таблицы ZOBJECT). Для просмотра файла KnowledgeC.db можно воспользоваться любой программой для работы с базой данных.

Файл netusage.sqlite [4] также является базой данных sqlite и располагается по пути /private/var/networkd или /private/var/networkd/db в зависимости от версии ОС. В базе данных хранится информация об использовании программами (процессами) сети: наименование процесса, использующего сеть (поле ZPROCNAME таблицы ZPROCESS), дата и время первого использования процессом сети (поле ZFIRSTTIMESTAMP таблицы ZPROCESS), дата и время последнего использования сети (поле ZTIMESTAMP таблицы ZPROCESS), количество полученных по сети пакетов (поле ZPACKETSIN таблицы ZLIVEROUTEPERF), байт (поле ZBYTESIN таблицы ZLIVEROUTEPERF), количество отправленных по сети пакетов (поле ZPACKETSOUT таблицы ZLIVEROUTEPERF), байт (поле ZBYTESOUT таблицы ZLIVEROUTEPERF). Для автоматизированного извлечения информации из netusage.sqlite можно воспользоваться программой mac_apt с плагином NETUSAGE, а для просмотра содержимого — любой программой для работы с базой данных.

Основная информация, которая интересует специалиста при анализе указанных массивов это: имена (полные пути до) файлов, временные метки запуска и

установки, дополнительные сведения о установщике, версии, источнике получения ПО.

IV. Массивы данных, содержащие информацию о существующих внутри раздела и открытых файлах

Файл com.apple.finder.plist [11] имеет формат plist и располагается по пути /Users/%username%/Library/Preferences. Для каждого каталога, который открывался во время работы системы в ключе FXRecentFolders хранится ключ, содержащий ключи name (имя каталога) и file-bookmark [12], в котором хранится полный путь к каталогу и дата создания каталога. Для автоматизированного извлечения информации из файла com.apple.finder.plist можно воспользоваться программой mac_apt с плагином RECENTITEMS, а для просмотра содержимого — программой plist Editor.

В каталоге /Users/%username%/Library/Application Support/com.apple.sharedfilelist расположены файлы с расширением sfl2 [11] Среди них стоит отметить com.apple.LSSharedFileList.RecentDocuments.sfl2 и RecentApplications.sfl2, хранящие списки недавно используемых приложений и списки недавно открытых файлов. Также в каталоге com.apple.sharedfilelist расположен подкаталог com.apple.LSSharedFileList.ApplicationRecentDocuments с файлами <домен>.<название разработчика>.<название приложения>.sfl2, в которых хранятся списки недавно открытых файлов приложением, имя которого указано в названии файла. Все вышеперечисленные файлы имеют структуру формата bplist. В них для каждого объекта, к которому недавно обращался пользователь, в ключе items хранится ключ, содержащий ключи uuid [13] (уникальный идентификатор объекта), Name (имя объекта) и Bookmark, в котором хранятся полный путь к объекту и дата изменения объекта. Для автоматизированного извлечения информации из этих файлов можно воспользоваться программой mac_apt с плагином RECENTITEMS, а для просмотра содержимого — программой plist Editor.

Каталог /.Spotlight-V100/Store-V2/<UUID> [14]–[16] содержит бинарные файлы store.db и .store.db, в которых сохраняются результаты индексирования файлов. Ручной анализ данных файлов является трудо- и время-затратной процедурой, поэтому для автоматизированного извлечения информации можно воспользоваться программой mac_apt с плагином SPOTLIGHT. В результате применения mac_apt информация из файлов собирается в единый массив в виде текстового файла (пример фрагмента файла представлен на рисунке 3). В формируемом массиве для каждого индексируемого файла содержатся Inode_Num (идентификатор файла), Parent_Inode_Num (идентификатор родительского каталога файла), Store_ID (номер записи файла в базе индексирования), Last_Updated (временная отметка последнего обновления индекса), kMDItemContentType (дополнительные сведения, например, тип содержимого). Анализировать все записи Spotlight не требуется – критерий фильтрации – исполняемые файлы (binary executable) и скрипты (script) по соответствующему свойству kMDItemContentType.

```
Inode_Num --> 3586532
Flags --> 32
Store_ID --> 74242
Parent_Inode_Num --> 3092740
Last_Updated --> 2023-08-24 15:29:04.615377
_kMDItemContentChangeDate --> 2023-03-04 22:26:10
_kMDItemContentCreationDateHour --> 3
_kMDItemContentCreationDateMonth --> 3
_kMDItemContentCreationDateWeekOfMonth --> 2
_kMDItemContentCreationDateWeekOfYear --> 10
_kMDItemContentCreationDateWeekday --> 1
_kMDItemContentCreationDateWeekdayOrdinal --> 1
_kMDItemContentCreationDateYear --> 2023
_kMDItemContentModificationDateHour --> 3
_kMDItemContentModificationDateMonth --> 3
_kMDItemContentModificationDateWeekOfMonth --> 2
_kMDItemContentModificationDateWeekOfYear --> 10
_kMDItemContentModificationDateWeekday --> 1
_kMDItemContentModificationDateWeekdayOrdinal --> 1
_kMDItemContentModificationDateYear --> 2023
_kMDItemCreationDate --> 2023-03-04 22:26:10
```

Рис. 3. Фрагмент содержимого файла, формируемого программой mac_apt с плагином SPOTLIGHT

В каталоге /.fseventsd [16] хранятся двоичные файлы, сжатые через GZIP, которые содержат полные пути к файлам и наименования произведенных над ними файловых операций. Распакованные файлы содержат заголовок, который начинается с сигнатуры (ASCII-строка «1SLD», «2SLD» или «3SLD»), и записи, состоящие из полей: полный путь к файлу, идентификатор файла и флаги файловых операций, по которым определяются наименования файловых операций. Для автоматизированного извлечения информации из данных файлов можно воспользоваться программой mac_apt с плагином FSEVENTS.

Файл com.apple.LaunchServices.QuarantineEventsV2 [17], [18] является базой данных sqlite и располагается по пути /Users/%username%/Library/Preferences. В базе данных хранится информация о файлах, помещенных в «карантин»: UUID записи в базе данных (поле LSQuarantineEventIdentifier), дата и время создания записи о файле в базе данных (поле LSQuarantineTimeStamp), название сервиса, поместившего файл в карантин (поле LSQuarantineAgentBundleIdentifier), название программы, поместившей файл в карантин (поле LSQuarantineAgentName). Для автоматизированного извлечения информации из файла com.apple.LaunchServices.QuarantineEventsV2 можно воспользоваться mac_apt с плагином QUARANTINE, а для просмотра содержимого — любой программой для работы с базой данных. Файлы в ОС macOS кроме атрибутов файловой системы имеют так называемые расширенные атрибуты, для просмотра которых используется команда xattr. У файлов, загруженных из сети Интернет, присутствует расширенный атрибут com.apple.quarantine (рисунок 4). Атрибут содержит дату и время занесения файла в карантин в формате Unix, имя приложения, которое инициировало загрузку файла и UUID записи, хранящийся в базе данных com.apple.LaunchServices.QuarantineEventsV2.

Рис. 4. Вывод расширенных атрибутов загруженных файлов

Основная информация, которая интересует специалиста при анализе указанных массивов это: имена (полные пути до) открытых файлов, в т.ч. (исполняемых), временные метки открытия, дополнительные сведения о ПО, с использованием которого открыты файлы.

V. Модель процесса выявления следов использования ПО

Расследование инцидента связано с риском искажения исходных данных вследствие ошибок в выполнении базовых процедур по доступу к данными и их извлечению, а также с неверной интерпретацией результатов. Для минимизации ошибок, связанных с т.н. «человеческим фактором» была представлена концепция о формализации процесса расследования инцидента ИБ. Так, в работах [19]–[22] авторами предлагаются обобщенные модели процесса расследования, в которых уделено внимание порядку выполнения основных процедур и документирования получаемых данных. Работы [23]–[25] описывают дополнительные (в т.ч. уточняющие) этапы расследования с разделением специалистов по защите информации в различные категории и указанием их обязанностей.

Вместе с тем, при формализации этапов расследования в указанных выше работах слабо акцентируются детали и особенности сбора и анализа данных, в т.ч. связанных с использованием ПО. Рассмотрим предлагаемую авторами модель *M*, целью которой является формализация процесса выявления следов использования ПО в macOS.

Модель *M* описывается кортежем:

$$M = \langle S, P, A, \Delta t, \Delta n, Q \rangle, \quad (1)$$

где:

– *S* — множество массивов данных, содержащих информацию о запускавшихся программах и открывавшихся (создававшихся) файлах;

– *P* — множество записей сопоставления имен открытых файлов с определенным ПО;

– *A* — множество средств автоматизации процедуры нормализации (под нормализацией понимается преобразование разноформатной информации к единому структурированному виду) данных из массивов множества *S*;

– Δt — значение временного интервала для объединения однотипных записей;

– Δn — значение метрики Левенштейна для объединения однотипных записей;

– *Q* — формируемая временная шкала следов использования ПО.

Одним из вариантов моделирования процесса является применение методологии IDEF0 [26]. Общий вид рассматриваемой модели представлен IDEF0-нотацией на рисунке 5.

Построение временной шкалы *Q*, выполняемое в блоке A0 модели, состоит из нескольких этапов, рассматриваемых далее.

Наличие свободно распространяемых средств автоматизации процедуры нормализации данных из ранее указанных массивов позволяет преобразовать интересующие специалиста данные к записи *r*, представленной кортежем:

$$r = \langle N, T, C, D \rangle, \quad (2)$$

где:

– *N* — имя/полный путь до файла;

– *T* — временная метка установки / запуска / открытия. Может отсутствовать в некоторых массивах данных;

– *C* — дополнительные сведения (заголовок окна программы, название установщика и пр.), позволяющие идентифицировать ПО / открытый файл;

– *D* — название массива данных *s*, откуда получена информация, $s \in S$.

Множество *R* разделяется подмножество R^* записей о запускавшихся программах, $r^* \in R^*$ и подмножество R' записей об открытых файлах, $r' \in R'$, $R = R^* \bigcup R'$.

Записи r' лишь косвенно свидетельствуют о следах использования ПО. Для отображения $r' \to r^*$, дополняя подмножество R^*, специалисту следует воспользоваться заранее подготовленными записями сопоставления имен открытых файлов с определенным ПО, представляемыми в виде кортежа *p*:

$$p = \langle N, C \rangle, \quad (3)$$

где: *N* — имя/полный путь ПО; *C* — дополнительные сведения, содержащие критерии сопоставления.

Критериями сопоставления являются, например, перечни имен характерных файлов и/или их расширений (в т.ч. в виде регулярных выражений), а также необходимое количество совпадений записей r' с перечнями — для уменьшения вероятности неверного сопоставления.

Этапы процесса, описываемые моделью *M*, представлены в виде IDEF0-нотации на рисунке 6.

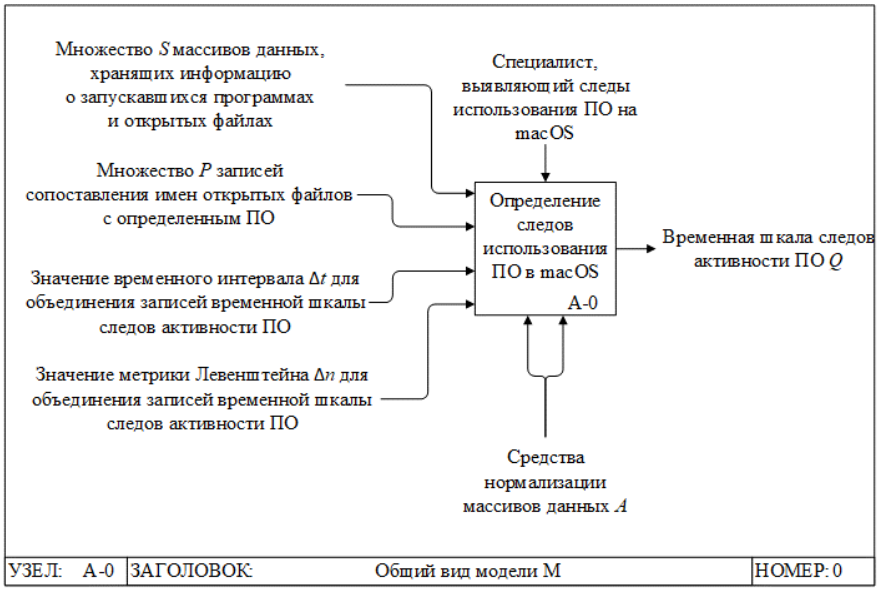

Рис. 5. Общий вид модели *M* процесса выявления следов использования ПО в macOS

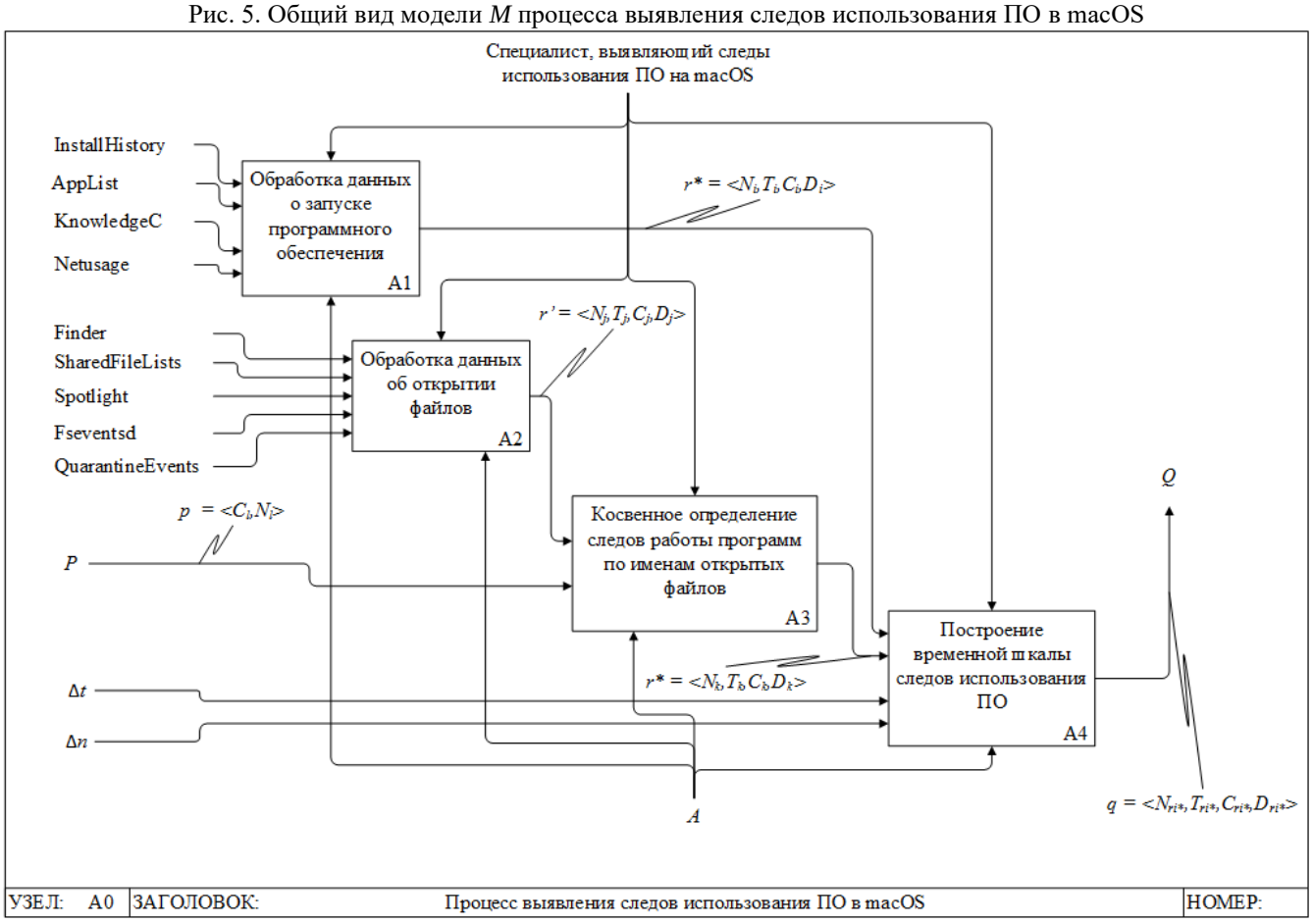

Рис. 6. Этапы процесса выделения следов использования ПО в macOS

Рассмотрим применение модели M, определение критериев сопоставления на примерах «троянов-шифровальщиков» (троянов) для macOS.

VI. Применение разработанной модели для выявления следов использования ПО

Первый «троян» называется «MacRansom» [27], отличительной особенностью которого является механизм шифрования по заданному времени, т.е. шифрование файлов начинается не сразу, а по заданному временному значению. При обнаружении указанных файлов до момента шифрования появляется возможность их удаления для сохранности пользовательских данных. Критерии сопоставления представлены в таблице 1. Для уменьшения вероятности ошибочного сопоставления оба имени должны присутствовать в массивах анализируемых данных.

Таблица 1. Критерии сопоставления *C* для трояна MacRansom

Тип данных	Значение	Путь до файла
Имя файла	.FS_Store	~/Library/
Имя файла	._README_	

Второй «троян» называется «EvilQuest» [28]. Критерии сопоставления представлены в таблице 2. Для уменьшения вероятности ошибочного сопоставления четыре имени файла, а также хотя бы одно имя файла с расширением «.locked» должны присутствовать в массивах анализируемых данных.

Таблица 2. Критерии сопоставления *C* для трояна EvilQuest

Тип данных	Значение	Путь до файла
Имя файла	toolroomd	~/Library/mixednkey/
Имя файла	com.apple.questd	~/Library/AppQuest/
Имя файла	com.apple.questd.plist	~/Library/LaunchAgents/
Имя файла	READ_ME_NOW.txt	~/Documents/
Расширение	.locked	

Третий «троян» называется «KeRanger» [29]. Механизм отложенного шифрования похож на «MacRansom», но временная задержка фиксирована на значении 72 часов. Критерии сопоставления представлены в таблице 3. Для уменьшения вероятности ошибочного сопоставления 6 имен файлов, а также хотя бы одно имя файла с расширением «.encrypted» должны присутствовать в массивах анализируемых данных.

Таблица 3. Критерии сопоставления *C* для трояна KeRanger

Тип данных	Значение	Путь до файла
Имя файла	Gereral.rtf	
Имя файла	.kernel_service	~/Library/
Имя файла	.kernel_pid	~/Library/
Имя файла	.kernel_time	~/Library/
Имя файла	.kernel_complete	~/Library/
Имя файла	README_FOR_DECRYPT.txt	
Расширение	.encrypted	

Время запуска ПО *T* необходимо зафиксировать на времени создания/открытия первого файла из записи r'.

Для формирования множества *Q*, состоящего из упорядоченных по времени записей *q* и представляющего собой временную шкалу, широко используемую при расследовании инцидентов ИБ [30], [31], необходимо произвести обработку записей r^* для их объединения по значению имени массива данных *D*.

Первый критерий объединения — значение интервала времени Δt, которое выбирается исходя из удобного масштаба отображения временной шкалы элементов множества *Q*. Экспериментально установлено, что время задержки между появлением информации о запуске ПО / открытии файла в различных массивах данных составляет до 2 секунд и зависит от нагрузки на дисковую подсистему компьютера, количества одновременно выполняемых ресурсозатратных задач и др. По-умолчанию значение Δt имеет смысл установить равным 2 секундам.

Второй критерий объединения — одинаковые (схожие) имена открытых файлов / запущенных программ. Для определения «похожести» текстовых строк применяют различные метрики [32], **Ошибка! Источник ссылки не найден.**, среди которых наиболее широкое распространение получили: поиска наибольшей общей подпоследовательности (НОП), метрики Левенштейна, Хэмминга и «q-грамм» [34]. В рассматриваемом случае, для сравнения имен файлов, была выбрана метрика Левенштейна Δn, т.к. она обладает следующими преимуществами [35]:

– большей универсальностью относительно поиска НОП за счет наличия операции замены;

– возможностью сравнения строк различной длины в отличие от метрики Хэмминга;

– меньшими затратами вычислительных ресурсов и памяти по сравнению с расчетом метрики «q-грамм».

Одинаковые имена файлов имеют метрику Δn равную 0, следовательно, Δn имеет смысл установить равным 0 для уменьшения вероятности появления ошибок 2 рода.

Исходя из двух критериев объединения и выражений (1) – (3) запись *q* описывается выражением:

$$\forall r_i^* \exists q_j \left\| \begin{cases} \left\langle N_{r_i^*}, T_{r_i^*}, C_{r_i^*}, D_{r_i^*} \right\rangle \left\| T_{r_{i+k}^*} - T_{r_i^*} > \Delta t, L\left(\left\langle N_{r_i^*}, ..., N_{r_{i+k}^*} \right\rangle\right) \neq \Delta n \right. \\ \left\langle N_{r_i^*}, T_{r_i^*}, C_{r_i^*}, \left\langle D_{r_i^*}, ..., D_{r_{i+k}^*} \right\rangle \right\rangle \left\| T_{r_{i+k}^*} - T_{r_i^*} \leq \Delta t, L\left(\left\langle N_{r_i^*}, ..., N_{r_{i+k}^*} \right\rangle\right) = \Delta n \end{cases} \right., \quad (4)$$

где *L* — функция расчета значения метрики Левенштейна.

Построение временной шкалы *Q* заключается в сортировке по возрастанию записей *q* по полю *T* и отображении их в виде точек на оси абсцисс *t*. Пример временной шкалы для использования вредоносного ПО «MacRansom» представлен на рисунке 7.

Приведенный на рис. 7 пример частично отображает временную шкалу операций над файлами (ФО), свидетельствующих об использовании вредоносного ПО «MacRansom». В качестве точки q_1 взят запуск исполняемого файла, после чего наблюдаются ФО по созданию файлов «._FS_Store» и «._README_» (точки q_3 и q_4). Точки q_{15} и q_{16} соответствуют ФО по изменению

содержимого пользовательских файлов — их шифрованию. ФО, как было сказано ранее, сохраняются в т.ч. в массиве «Fseventsd», в котором не фиксируется время, но записи сохраняются в порядке совершения ФО в ОС, что позволяет их соотнести с точками на шкале времени.

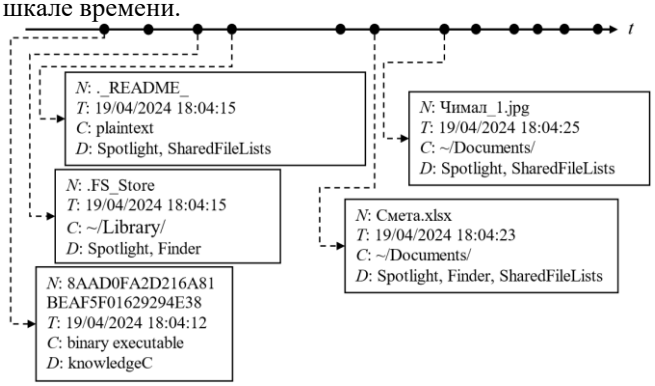

Рис. 7. Временная шкала Q следов использования вредоносного ПО «MacRansom» в macOS

VII. Заключение

Рассмотренная в статье модель позволяет формализовать процесс анализа интересующих специалиста сведений об использовании ПО в ОС macOS в рамках расследования инцидента ИБ. Порядок анализа описанных массивов данных позволяет уменьшить ошибки, связанные с т.н. «человеческим фактором», а также ускорить процедуру расследования инцидента за счет представления событий в виде временной шкалы

Благодарности

Авторы выражают благодарность Поршневу С.В. за помощь в структурировании текста статьи и уточнении формулировок.

Библиография

[1] (Desktop operating system market share worldwide) [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide.

[2] (Complete list of Mac viruses, malware and trojans) [Online]. Available: https://www.macworld.com/article/672879/list-of-mac-viruses-malware-and-security-flaws.html.

[3] K. Kent, S. Chevalier, T. Grance. Guide to integrating forensic techniques into incident response. NIST Special Publication, vol. 10, pp. 800–886, 2006.

[4] J. Bradley. OS X incident response: scripting and analysis. Rockland: Syngress Publishing, Inc., 2016. https://doi.org/10.1016/C2015-0-00440-3.

[5] G. Johansen. Digital forensics and incident response: incident response techniques and procedures to respond to modern cyber threats, 2nd Edition, Birmingham: Packt Publishing Ltd., 2020.

[6] J. T. Luttgens, M. Pepe, K. Mandia. Incident response & computer forensics, third edition, New York: McGraw Hill Ltd., 2014.

[7] M. H. Ligh, J. Levy. The art of memory forensics: detecting malware and threats in Windows, Linux, and Mac memory, New York: John Wiley & Sons, Inc., 2014.

[8] J. Levin. Mac OS X and iOS internals to the Apple’s core. New York: John Wiley & Sons, Inc., 2013.

[9] (GitHub — ydkhatri/mac_apt: macOS (& ios) artifact parsing tool) [Online]. Available: https://github.com/ydkhatri/mac_apt.

[10] (Knowledge is power! using the macOS/iOS knowledgeC.db database to determine precise user and application usage) [Online]. Available: https://www.mac4n6.com/blog/2018/8/5/knowledge-is-power-using-the-knowledgecdb-database-on-macos-and-ios-to-determine-precise-user-and-application-usage.

[11] P. Shukla, A. Pratap Forensic investigation: Apple devices acquisition & analysis. *NFSU Journal of Cybersecurity & Digital Forensics*, vol. 1, no. 1, pp. 17-24, 2022.

[12] (Mac Bookmark Format — mac_alias 2.2.2 documentation) [Online]. Available: https://mac-alias.readthedocs.io/en/latest/bookmark_fmt.html.

[13] (RFC 9562 — Universally Unique Identifiers) [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9562.

[14] S. T. Atwal, M. Scanlon, N. Le-Khac. Shining a light on Spotlight: leveraging Apple's desktop search utility to recover deleted file metadata on macOS. *Digital Investigation*, vol. 28, pp. S105-S115, 2019. https://doi.org/10.1016/j.diin.2019.01.019.

[15] Y. Khatri. Investigating spotlight internals to extract metadata. *Digital Investigation*, vol. 28, pp. S96-S103, 2019. https://doi.org/10.1016/j.diin.2019.01.005.

[16] J. Joun, S. Lee, J. Park. Discovering spoliation of evidence through identifying traces on deleted files in macOS. *Digital Investigation*, vol. 44, 2023. https://doi.org/10.1016/j.fsidi.2023.301502.

[17] B. Maddu, R. Maddu. OS X artifact analysis. *International Journal of Recent Technology and Engineering*, vol. 7, pp. 26-32, 2019.

[18] R. Niranjan. Mac OS forensics. *Practical Cyber Forensics*, vol. 1, pp. 101-132, 2019. https://doi.org/10.1007/978-1-4842-4460-9_4.

[19] V. Baryamureeba, F. Tushabe. The enhanced digital investigation process model. *Proceedings of the Fourth Digital Forensic Research Workshop*, vol. 1, pp. 1–9, 2004.

[20] F. C. Freiling, B. Schwittay. A common process model for incident response and computer forensics. *International Conference on IT-Incidents Management & IT-Forensics*, vol. 1, pp. 19–39, 2007.

[21] A. Agarwal, M. Gupta, S. Gupta. Systematic digital forensic investigation model. *International Journal of Computer Science and Security (IJCSS)*, vol. 5, no. 1, pp. 118-131, 2011.

[22] M. D. Kohn, M. M. Eloff, J. H. P. Eloff. Integrated digital forensic process model. *Computers & Security*, vol. 38, pp. 103–115, 2013. https://doi.org/10.1016/j.cose.2013.05.001.

[23] L. Johnson. Computer incident response and forensics team management. Rockland: Syngress Publishing, Inc.; 2014. https://doi.org/10.1016/C2012-0-01092-7.

[24] D. Lillis, B. Becker, T. O'Sullivan. Current challenges and future research areas for digital forensic investigation. *Proceedings of 11th ADFSL Conference on Digital Forensics, Security and Law (CDFSL 2016)*, vol. 1, pp. 9–20, 2016. https://doi.org/10.48550/arXiv.1604.03850.

[25] X. Du, N. Le-Khac, M. Scanlon. Evaluation of digital forensic process models with respect to digital forensics as a service. *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS 2017)*, vol. 1, pp. 573–581, 2017.

[26] A. Presley, D. H. Liles. The use of IDEF0 for the design and specification of methodologies, vol. 1, pp. 5–12, 1998.

[27] (MacRansom offered as ransomware as a service) [Online]. Available: https://www.fortinet.com/blog/threat-research/macransom-offered-as-ransomware-as-a-service.

[28] (EvilQuest ransomware mac) [Online]. Available: https://www.pcrisk.com/removal-guides/18425-evilquest-ransomware-mac.

[29] (New OS X ransomware KeRanger infected Transmission bittorrent client installer) [Online]. Available: https://unit42.paloaltonetworks.com/new-os-x-ransomware-keranger-infected-transmission-bittorrent-client-installer.

[30] T. J. Grant, E. Eijk, H. S. Venter. Assessing the feasibility of conducting the digital forensics process in real time. *Proceedings of 11th International Conference on Cyber Warfare & Security (ICCWS 2016)*, vol. 25, pp. 147–155, 2016.

[31] M. Debinski, F. Breitinger, P. Mohan. Timeline2GUI: A Log2Timeline CSV parser and training scenarios. *Digital Investigation*, vol. 28, pp.34–43, 2019. https://doi.org/10.1016/j.diin.2018.12.004.

[32] M. M. Deza, E. Deza. Encyclopedia of distances, 2nd edition. Heidelberg: Springer Berlin, 2013. https://doi.org/10.1007/978-3-642-30958-8.

[33] D. Gusfield. Algorithms on strings, trees and sequences: computer science and computational biology, 1st edition, Cambridge: Cambridge University Press; 1997.

[34] E. Sutinen, J. Tarhio. Approximate string matching with ordered q-grams. *Nordic Journal of Computing*, vol. 11, no. 4, pp. 321–343, 2004.

[35] G. Navarro. A guided tour to approximate string matching. *ACM Computting Surveys, vol. 33, no. 1, pp. 31–88, 2001. https://doi.org/10.1145/375360.375365.*

Software usage artifacts identification process model on macOS operating systems used in information security incidents investigation

Roman V. Gibilinda, Nataliya S. Knyazeva, and Ksenia S. Semenova

***Abstract* — The article presents a software usage artifacts detecting process model used in an information security incidents investigation on macOS operating systems. The emphasis is on examples of identifying malicious software running artifacts. The input data of the pro-posed model are data arrays containing features related to the fact of a program activity: the name (full path to) the file, the launch time and the data source about the activity. The article briefly describes structure of data arrays and shows indirect feature software identification process based on accessing and creating files. The proposed model can be expanded by additional (new) data arrays, and can also be used as the basis for creating in-formation analysis software to speed up the information security incident response process.**

***Keywords* — information security incident response, software, artifacts, IDEF0, macOS.**

REFERENCES

[1] (Desktop operating system market share worldwide) [Online]. Available: https://gs.statcounter.com/os-market-share/desktop/worldwide.

[2] (Complete list of Mac viruses, malware and trojans) [Online]. Available: https://www.macworld.com/article/672879/list-of-mac-viruses-malware-and-security-flaws.html.

[3] K. Kent, S. Chevalier, T. Grance. Guide to integrating forensic techniques into incident response. NIST Special Publication, vol. 10, pp. 800–886, 2006.

[4] J. Bradley. OS X incident response: scripting and analysis. Rockland: Syngress Publishing, Inc., 2016. https://doi.org/10.1016/C2015-0-00440-3.

[5] G. Johansen. Digital forensics and incident response: incident response techniques and procedures to respond to modern cyber threats, 2nd Edition, Birmingham: Packt Publishing Ltd., 2020.

[6] J. T. Luttgens, M. Pepe, K. Mandia. Incident response & computer forensics, third edition, New York: McGraw Hill Ltd., 2014.

[7] M. H. Ligh, J. Levy. The art of memory forensics: detecting malware and threats in Windows, Linux, and Mac memory, New York: John Wiley & Sons, Inc., 2014.

[8] J. Levin. Mac OS X and iOS internals to the Apple's core. New York: John Wiley & Sons, Inc., 2013.

[9] (GitHub — ydkhatri/mac_apt: macOS (& ios) artifact parsing tool) [Online]. Available: https://github.com/ydkhatri/mac_apt.

[10] (Knowledge is power! using the macOS/iOS knowledgeC.db database to determine precise user and application usage) [Online]. Available: https://www.mac4n6.com/blog/2018/8/5/knowledge-is-power-using-the-knowledgecdb-database-on-macos-and-ios-to-determine-precise-user-and-application-usage.

[11] P. Shukla, A. Pratap Forensic investigation: Apple devices acquisition & analysis. *NFSU Journal of Cybersecurity & Digital Forensics*, vol. 1, no. 1, pp. 17-24, 2022.

[12] (Mac Bookmark Format — mac_alias 2.2.2 documentation) [Online]. Available: https://mac-alias.readthedocs.io/en/latest/bookmark_fmt.html.

[13] (RFC 9562 — Universally Unique Identifiers) [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9562.

[14] S. T. Atwal, M. Scanlon, N. Le-Khac. Shining a light on Spotlight: leveraging Apple's desktop search utility to recover deleted file metadata on macOS. *Digital Investigation*, vol. 28, pp. S105-S115, 2019. https://doi.org/10.1016/j.diin.2019.01.019.

[15] Y. Khatri. Investigating spotlight internals to extract metadata. *Digital Investigation*, vol. 28, pp. S96-S103, 2019. https://doi.org/10.1016/j.diin.2019.01.005.

[16] J. Joun, S. Lee, J. Park. Discovering spoliation of evidence through identifying traces on deleted files in macOS. *Digital Investigation*, vol. 44, 2023. https://doi.org/10.1016/j.fsidi.2023.301502.

[17] B. Maddu, R. Maddu. OS X artifact analysis. *International Journal of Recent Technology and Engineering*, vol. 7, pp. 26-32, 2019.

[18] R. Niranjan. Mac OS forensics. *Practical Cyber Forensics*, vol. 1, pp. 101-132, 2019. https://doi.org/10.1007/978-1-4842-4460-9_4.

[19] V. Baryamureeba, F. Tushabe. The enhanced digital investigation process model. *Proceedings of the Fourth Digital Forensic Research Workshop*, vol. 1, pp. 1–9, 2004.

[20] F. C. Freiling, B. Schwittay. A common process model for incident response and computer forensics. *International Conference on IT-Incidents Management & IT-Forensics*, vol. 1, pp. 19–39, 2007.

[21] A. Agarwal, M. Gupta, S. Gupta. Systematic digital forensic investigation model. *International Journal of Computer Science and Security (IJCSS)*, vol. 5, no. 1, pp. 118-131, 2011.

[22] M. D. Kohn, M. M. Eloff, J. H. P. Eloff. Integrated digital forensic process model. *Computers & Security*, vol. 38, pp. 103–115, 2013. https://doi.org/10.1016/j.cose.2013.05.001.

[23] L. Johnson. Computer incident response and forensics team management. Rockland: Syngress Publishing, Inc.; 2014. https://doi.org/10.1016/C2012-0-01092-7.

[24] D. Lillis, B. Becker, T. O'Sullivan. Current challenges and future research areas for digital forensic investigation. *Proceedings of 11th ADFSL Conference on Digital Forensics, Security and Law (CDFSL 2016)*, vol. 1, pp. 9–20, 2016. https://doi.org/10.48550/arXiv.1604.03850.

[25] X. Du, N. Le-Khac, M. Scanlon. Evaluation of digital forensic process models with respect to digital forensics as a service. *Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS 2017)*, vol. 1, pp. 573–581, 2017.

[26] A. Presley, D. H. Liles. The use of IDEF0 for the design and specification of methodologies, vol. 1, pp. 5–12, 1998.

[27] (MacRansom offered as ransomware as a service) [Online]. Available: https://www.fortinet.com/blog/threat-research/macransom-offered-as-ransomware-as-a-service.

[28] (EvilQuest ransomware mac) [Online]. Available: https://www.pcrisk.com/removal-guides/18425-evilquest-ransomware-mac.

[29] (New OS X ransomware KeRanger infected Transmission bittorrent client installer) [Online]. Available: https://unit42.paloaltonetworks.com/new-os-x-ransomware-keranger-infected-transmission-bittorrent-client-installer.

[30] T. J. Grant, E. Eijk, H. S. Venter. Assessing the feasibility of conducting the digital forensics process in real time. *Proceedings of 11th International Conference on Cyber Warfare & Security (ICCWS 2016)*, vol. 25, pp. 147–155, 2016.

[31] M. Debinski, F. Breitinger, P. Mohan. Timeline2GUI: A Log2Timeline CSV parser and training scenarios. *Digital Investigation*, vol. 28, pp.34–43, 2019. https://doi.org/10.1016/j.diin.2018.12.004.

[32] M. M. Deza, E. Deza. Encyclopedia of distances, 2nd edition. Heidelberg: Springer Berlin, 2013. https://doi.org/10.1007/978-3-642-30958-8.

[33] D. Gusfield. Algorithms on strings, trees and sequences: computer science and computational biology, 1st edition, Cambridge: Cambridge University Press; 1997.

[34] E. Sutinen, J. Tarhio. Approximate string matching with ordered q-grams. *Nordic Journal of Computing*, vol. 11, no. 4, pp. 321–343, 2004.

[35] G. Navarro. A guided tour to approximate string matching. *ACM Computting Surveys*, vol. 33, no. 1, pp. 31–88, 2001. https://doi.org/10.1145/375360.375365.