В.А. Ковязин, О.И. Пинигина, Т.И. Паюсова

Тюменский государственный университет, г. Тюмень

УДК 004.49

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ САМОМОДИФИЦИРУЮЩЕГОСЯ КОДА В ОБЛАСТИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Аннотация. В статье рассматриваются возможности самомодифицирующегося кода и виды вирусов, принцип работы которых основан на нем, а также методы защиты от данных вирусов.

Ключевые слова: самомодифицирующийся код, полиморфные вирусы, метаморфные вирусы, проактивная защита.

Введение

Согласно исследованию международной ИТ-компании Positive Technologies [1], зачастую хакеры используют для реализации и сокрытия атак инструменты, основанные на самомодифицирующемся коде.

Целью работы является исследование возможностей самомодифицирующегося кода в области информационной безопасности.

В статье рассмотрен самомодифицирующийся код и сферы его применения; вирусы и пакеры, основывающиеся на нем, а также способы защиты от них.

Самомодифицирующийся код

Самомодифицирующийся код – возможность программы создавать, изменять или удалять части своего кода во время выполнения. Обычно применение данной возможности используется в приложениях, написанных для работы с процессорами, которые не разделяют память для кода и память для данных. Такая архитектура применяется в процессорах с архитектурой фон Неймана. Главное преимущества использования самомодификации – возможность избежать ветвления программы из-за большого количества

проверок условий и установки различных флагов состояний программы. Термин применим только к коду, который изменяет себя по заранее продуманному алгоритму, а не в результате ошибки, например, переполнения стека (Stack Overflow) [2].

Метод самомодификации делится на 2 типа в зависимости от времени ее проведения:

1) во время инициализации – модификация выполняется один раз перед запуском модифицируемого кода или при распаковке из контейнера;

2) во время исполнения – модификация выполняется непосредственно во время работы программы.

В большинстве случаев изменяется машинный код программы. Создаются новые инструкции и вставляются в код вместо старых. Например, безусловный переход «JMP» заменяет собой условный переход «JZ». Также в архитектурах IBM/360 и Z/Architecture существует способ модификации с использованием стандартной инструкции «EX».

Метод самомодификации используется:

1) для препятствования исследования кода (полиморфные вирусы, метаморфные вирусы, пакеры и т. д.);

2) для повышения производительности программы;

3) для добавления отладочной функциональности;

4) для отключения частей ядра ОС, не используемых в данном аппаратном обеспечении;

5) для препятствования обратному проектированию кода с использованием отладчика;

6) для оптимизации кода, зависящего от состояния.

В интерпретируемых языках отсутствует возможность напрямую изменять существующий код, однако возможно создавать новый код во время исполнения программы, а также выполнять его используя методы языка.

Модификация также возможна за счет изменения указателя на вызываемую функцию во время работы программы.

Возможность самомодификации программы во время исполнения кода позволяет рассматривать программу, как виртуальную машину. Метапрограммирование позволяет перенести функции самомодификации в высокоуровневые языки, изменяя не непосредственно машинный код, а конструкции языка.

Выделяют несколько основных методов реализации:

- гомоиконность – код и инструкции по изменению кода основываются на одном синтаксисе.
- интроспекция – представление внутренних типов и структур языка в виде встроенных объектов, доступных для использования [3];
- выполнение произвольного кода, содержащегося в строковой переменной. Существует в большинстве интерпретируемых языков, как важная их часть.

Листинг 1. Код без применения самомодификации

```
for i in range(N):
    if should_increase is True:
        value = increase(value)
    else:
        value = decrease(value)
    process(value)
    if value > 30:
        should_increase = False
    elif value < 10:
        should_increase = True
```

Листинг 2. Код с применением самомодификации

```
for i in range(N):
    value = change(value)
    process(value)
    if value > 30:
        change = decrease
    elif value < 10:
        change = increase
```

Из данных примеров (листинги 1 и 2) можно установить, что самомодификация позволяет сократить количество кода, а также время его выполнения за счет пропуска проверки флага условия.

Полиморфные вирусы

Полиморфный вирус – это вирус, который шифрует себя, используя случайные пароли и различные способы шифрования; при заражении файлов, системных областей на диске изменяющий свой собственный код.

Вирус-полиморфик не содержит ни единого постоянного фрагмента кода. Благодаря шифрованию основного тела вируса и значительного изменения от одной копии к другой модуля-расшифровщика, для двух образцов одного и того же вируса совпадения будут отсутствовать.

Могут использоваться разнотипные методы шифрования [4], но чаще всего существует отдельная зеркальная пара для каждой операции. Это очень просто реализовать в ассемблере, где таких пар может быть много (например, ROL/ROR). Чрезвычайно важная особенность полиморфного вируса связана с тем, что вирус содержит специальные операнды, функции и процедуры, которые добавляются исключительно для становления кода более запутанным.

Полиморфный вирус нередко включает код, выполняемый в зависимости от конкретной ситуации. Так, при обнаружении вируса он может вызвать процедуру самоуничтожения как себя самого (непоправимая модификация кода), так и системы (массовое заражение системных файлов без возможности восстановления).

Первый в истории вирус-полиморфик, Chameleon, добавлял свой код в конец COM-файлов, используя два алгоритма шифрования. Один, в зависимости от значения ключа, шифровал тело по таймеру, а второй затруднял обнаружение вируса с помощью динамического кодирования.

Mutation Engine – первый генератор полиморфных вирусов для MS-DOS. Крайне сложен: в полиморфном декодере в любом количестве и порядке используются команды ROL, ROR, ADD, XOR, SUB. Больше половины инструкций 8086 можно использовать в командах для изменения параметров

шифрования, и, кроме того, можно использовать любые возможные способы адресации данных. В результате копии одного и того же вируса, зашифрованные с помощью MtE, при сопоставлении не совпадут ни одним байтом.

Создание MtE вскоре привело к появлению ряда других полиморфных генераторов, которые стали применять зеркальные функции, что значительно затруднило обнаружение вируса. Классические антивирусные базы и методы детектирования практически не дают желаемого эффекта при обнаружении вирусов-полиморфиков. Для их обнаружения необходимо совместить методы эвристического анализа и эмуляции.

Метаморфные вирусы

Метаморфный вирус – это вирус, который способен создавать новые копии, используя свою способность изменять, переводить и редактировать собственный код.

В метаморфных вирусах помимо декодера мутирует и весь исполняемый код, при этом за счет мутации отпадает необходимость их шифрования. В метаморфных вирусах неизменными являются только данные, которые при этом остается возможным замаскировать или сокрыть. В памяти не будет формироваться ничего, что оставалось бы постоянным от одного образца вируса к другому, поэтому детектировать метаморфный код с помощью метода эмуляции невозможно.

С точки зрения реализации существует два главных подхода к технологии мутации вирусов [5]:

- код вируса каждый раз «переписывается» с использованием инструкций фиксированной длины команд или встроенного дизассемблера, возвращающего как минимум длину инструкции;
- новый код создается каждый раз из определенного «прототипа», который хранится в зашифрованном виде.

Одна из существующих технологий трансформации метаморфных вирусов связана со вставкой и устранением из кода «мусора», что не воздействует на работу программы, но при этом затрудняет анализ больших блоков кода.

Иной способ трансформации метаморфных вирусов основан на воздействии на стандартные инструкции на уровне кода операции, подразумевающим переключение между несколькими отличающимися друг от друга кодами, выполняющими одинаковые функции.

Сложнейшей из представленных трансформацией метаморфного вируса представляется замена частей кода на функционально-эквивалентные. Для примера рассмотрим умножение числа x на три, которое возможно выразить как «x*3». Но альтернативным его представлением его можно назвать сумму трех x: «x+x+x». И первое, и второе выражения дают одинаковый результат, однако их внешнее представление отличается.

Крипторы

Создавать новые инструменты для каждой атаки слишком ресурсозатратно, поэтому многие группировки хакеров используют разработанное ПО несколько раз, что приводит к его обнаружению различными антивирусными компаниями, что в следствие снижает его эффективность его использования.

Для решения данной проблемы хакеры применяют специализированные инструменты – «крипторы», предназначенные для сокрытия вирусного ПО от детектирования антивирусами. Они упаковывают и шифруют код, а также изменяют его посредством мутаций.

До конца 2020 года регулярно проводились фишинговые рассылки, содержащие RTM (рис. 1). Этот процесс, по всей видимости, происходил автоматически.

Файлы, содержащиеся в письмах, сильно отличались в разных письмах (рис. 2), однако полезная нагрузка вредоносного ПО оставалась почти идентичной.

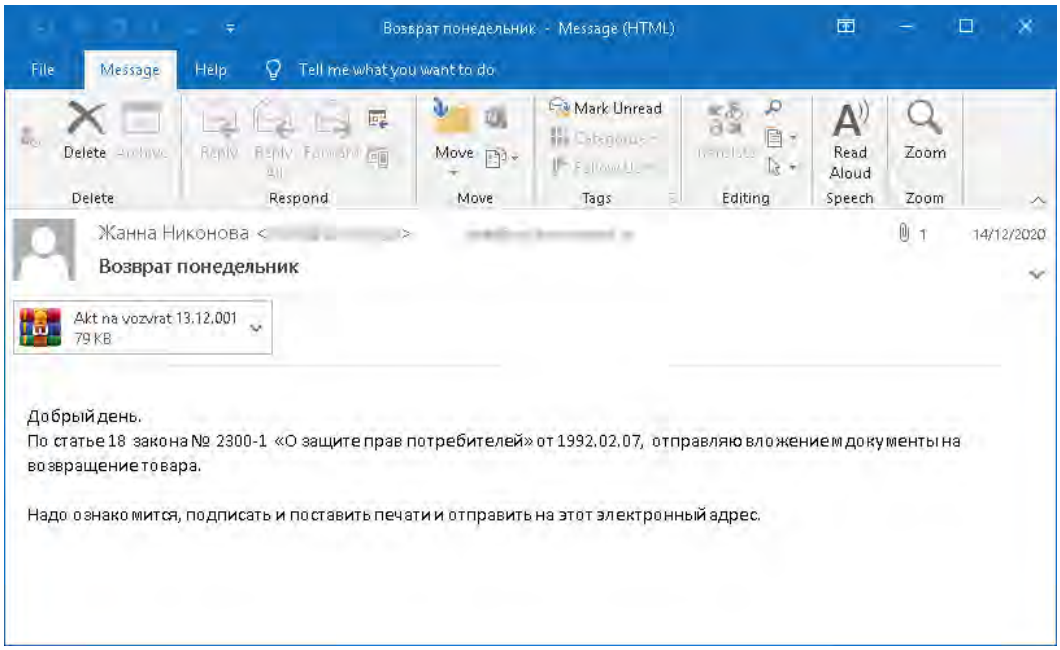

Рис. 1. Фишинговое письмо RTM, декабрь 2020

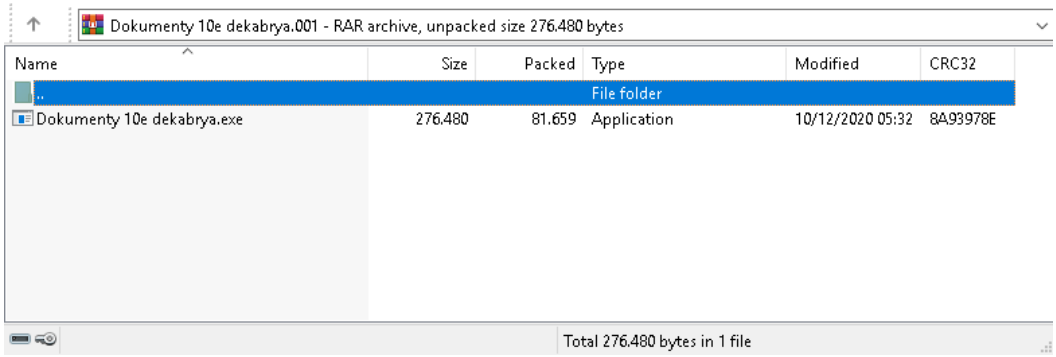

Рис. 2. Пример архива RTM

Исследование образцов, которые были упакованы более новыми алгоритмами, позволило обнаружить большое количество другого вирусного ПО, защищенного таким же способом. Использование похожих или аналогичных способов упаковки вредоносного ПО позволяет говорить о распространении сервисов типа Packer-as-a-Service, которые предоставляют услуги упаковки программного кода.

Проактивные методы антивирусной защиты

Проактивные методы защиты – методы защиты, направленные на предотвращение заражения, а не на поиск уже известных вирусов в системе. Они появились достаточно давно и начали развиваться совместно с реактивными методами, однако изначально они были слишком сложными в использовании,

поэтому их разработка была остановлена. Позже из-за распространения вирусов стало понятно, что только сигнатурных методов защиты недостаточно и сегмент проактивной защиты стал снова развиваться. В настоящий момент проактивные решения могут быть встраиваемыми в существующее антивирусное ПО или быть самостоятельным ПО, которое комбинирует различные методы проактивной защиты, эффективные против большого количества угроз.

Существуют следующие технологии проактивной защиты [6].

- Эвристический анализ – технология, позволяющая проводить анализ кода программы и выявлять потенциально вредоносные участки.
- Эмуляция кода – технология, позволяющая запускать программу в виртуальном окружении. Действия вредоносной программы не способны навредить системе, однако будут обнаружены и логированы.
- Анализ поведения – технология, которая перехватывает, обрабатывает и отслеживает системные события, также способна оценивать цепочки событий, вызываемых программами.
- Песочница – технология, которая ограничивает возможности исследуемой программы, чтобы она не могла повлиять на систему. Программы запускаются в изолированной среде, которая не дает доступ к важным системным функциям.
- Виртуализация рабочего приложения – технология, которая запросы на запись от приложения и записывает данные на виртуальный жесткий диск, очистка которого происходит при выключении питания компьютера.

СПИСОК ЛИТЕРАТУРЫ

1. PaaS, или Как хакеры ускользают от антивирусов – Текст: электронный // TAdviser – портал выбора технологий и поставщиков: [сайт]. – 2021. – URL: https://www.ptsecurity.com/ru-ru/research/pt-esc-threat-intelligence/ paas-or-how-hackers-evade-antivirus-software/ (дата обращения: 10.04.2021).

2. Self-modifying code – Текст: электронный // Wikipedia, the free encyclopedia: [сайт]. – URL: https://en.wikipedia.org/wiki/Self-modifying_code (дата обращения: 17.04.2021).

3. Малакшинов Б.С., Данилова С.Д. Применение метода самомодифицирующегося кода к автоматическому построению программного кода в онтологии с активной семантикой – Текст: электронный // Вестник БГУ. Математика, информатика. 2011. № 9 – URL: https://cyberleninka.ru/article/n/primenenie-metoda-samomodifitsiruyuschegosya-koda-k-avtomaticheskomu-postroeniyu-programmnogo-koda-v-ontologii-s-aktivnoy-semantikoy (дата обращения: 19.04.2021).

4. Долженков А.А., Грачева Е.В. Полиморфные вирусы – Текст: электронный // Успехи современного естествознания. – 2011. – № 7. – С. 105-105 – URL: http://www.natural-sciences.ru/ru/article/view?id=27093 (дата обращения: 14.04.2021).

5. Збицкий П.В. Модель метаморфного преобразования исполняемого кода – Текст: электронный // Вестник ЮУрГУ. Серия: Компьютерные технологии, управление, радиоэлектроника. 2009. № 26 (159). – URL: https://cyberleninka.ru/article/n/model-metamorfnogo-preobrazovaniya-ispolnyaemogo-koda (дата обращения: 16.04.2021).

6. Алиев А.Т. Проактивные системы защиты от вредоносного программного обеспечения – Текст: электронный // Известия ЮФУ. Технические науки. 2014. №2 (151) – URL: https://cyberleninka.ru/article/n/proaktivnye-sistemy-zaschity-ot-vredonosnogo-programmnogo-obespecheniya (дата обращения: 20.04.2021).