УДК 004.056 **DOI** 10.15622/ia.20.4.3

Д.П. Зегжда, М.О. Калинин, В.М. Крундышев, Д.С. Лаврова,
Д.А. Москвин, Е.Ю. Павленко

ПРИМЕНЕНИЕ АЛГОРИТМОВ БИОИНФОРМАТИКИ ДЛЯ ОБНАРУЖЕНИЯ МУТИРУЮЩИХ КИБЕРАТАК

Зегжда Д.П., Калинин М.О., Крундышев В.М., Лаврова Д.С., Москвин Д.А., Павленко Е.Ю. **Применение алгоритмов биоинформатики для обнаружения мутирующих кибератак.**

Аннотация. Функционал любой системы может быть представлен в виде совокупности команд, которые приводят к изменению состояния системы. Задача обнаружения атаки для сигнатурных систем обнаружения вторжений эквивалентна сопоставлению последовательностей команд, выполняемых защищаемой системой, с известными сигнатурами атак. Различные мутации в векторах атак (включая замену команд на равносильные, перестановку команд и их блоков, добавление мусорных и пустых команд) снижают эффективность и точность обнаружения вторжений. В статье проанализированы существующие решения в области биоинформатики, рассмотрена их применимость для идентификации мутирующих атак. Предложен новый подход к обнаружению атак на основе технологии суффиксных деревьев, используемой при сборке и проверке схожести геномных последовательностей. Применение алгоритмов биоинформатики позволяет добиться высокой точности обнаружения мутирующих атак на уровне современных систем обнаружения вторжений (более 90%), при этом превосходя их по экономичности использования памяти, быстродействию и устойчивости к изменениям векторов атак. Для улучшения показателей точности проведен ряд модификаций разработанного решения, вследствие которых точность обнаружения атак увеличена до 95% при уровне мутаций в последовательности до 10%. Метод может применяться для обнаружения вторжений как в классических компьютерных сетях, так и в современных реконфигурируемых сетевых инфраструктурах с ограниченными ресурсами (Интернет вещей, сети киберфизических объектов, сенсорные сети).

Ключевые слова: алгоритм Укконена, безопасность, биоинформатика, выравнивание, мутация, обнаружение вторжений, полиморфизм, сигнатура, суффиксное дерево.

1. Введение. Современные системы обнаружения вторжений (СОВ) выполняют функцию пассивной защиты отдельных хостов сети (хостовые СОВ) или сетевых соединений (сетевые СОВ), ведя наблюдение и анализ событий безопасности, происходящих внутри системы на уровне системных вызовов или сетевых потоков [1]. СОВ фиксируют признаки различных видов вредоносной активности, такие как эксплуатация программных уязвимостей, DDoS-атаки, сканирование портов и попытки проникновения в сеть. Независимо от типа к СОВ предъявляются высокие требования по качеству обнаружения и скорости распознавания атак, так как от этих характеристик зависит наносимый атакой урон [2, 3].

СОВ постоянно собирают различную информацию и составляют набор идентифицирующих признаков, анализируя который делают вы-

вод о текущей безопасности контролируемой системы. Распространенные сигнатурные СОВ для выявления вторжений сравнивают текущее состояние системы с известными шаблонами (сигнатурами) небезопасных состояний. Если текущее состояние совпадает с одной из сигнатур, СОВ сигнализируют о выявленном вторжении. В отличие от СОВ, построенных на основе оценки аномалий, в которых определяют отклонения от среднестатистического профиля поведения контролируемой системы, сигнатурные СОВ имеют высокие показатели качества и скорости обнаружения, низкий уровень ошибок первого и второго рода и не требуют выделенного этапа профилирования [4-6].

Определим защищаемую систему как набор сущностей, которые взаимодействуют друг с другом. Информационное взаимодействие между сущностями в системе реализуется путем выполнения команд. В результате взаимодействия изменяется состояние системы. Знание о том, что в интервале времени были в некоторой последовательности выполнены команды, определяет все состояния системы, получаемые из исходного. Поведение, которое представлено в виде последовательности команд, приводящих защищаемую систему в небезопасное состояние, будем называть атакой. Задача обнаружения вторжений для сигнатурной СОВ эквивалентна сопоставлению текущих последовательностей команд с известными сигнатурами атак, представленными последовательностями команд, ведущих к небезопасному состоянию. При сопоставлении последовательностей проблему, характерную для сигнатурного подхода, представляет наличие мутаций, или полиморфизм, атак [7-9]. Мутации усложняют обнаружение вторжений, поскольку требуют поддержания чрезмерно больших баз сигнатур для всех возможных вариаций атак, чрезмерных ресурсозатрат на поиск и сопоставление сигнатур с наблюдаемыми последовательностями, своевременного и постоянного пополнения сигнатур атак [10].

Мутации атак включают такие изменения в последовательности команд, как: замена команд на равносильные, перестановка команд и их блоков в цепочке, добавление мусорных и пустых команд. Например, мутирующие последовательности для цепочки $S = abcdefgh$: $S_1 = aBcdefgh$ (замена равносильной командой), $S_2 = abcedfgh$ (перестановка команд), $S_3 = abc\_edfgh$ (добавление мусорных команд, включая пустые команды и пропуски во времени). Поскольку для сигнатурных СОВ единственный способ определения атаки – нахождение идентичной последовательности в списке сигнатур, то даже незначительные изменения в последовательностях команд не позволяют сигнатурной СОВ распознать мутирующие атаки с использованием шаблонов.

Таким образом, сигнатурные СОВ уязвимы к полиморфизму атак и зависимы от наполнения базы сигнатур. Решением данной проблемы может служить быстродействующий механизм, способный с высокой точностью обрабатывать и сопоставлять поступающие последовательности команд с большим набором имеющихся сигнатур, несмотря на мутации в текущей последовательности. Для решения поставленной задачи в статье рассматриваются следующие аспекты:

– проанализированы существующие алгоритмы биоинформатики, которые позволяют решить аналогичную природную задачу – локализовать совпадения между последовательностью биокодов о состоянии биологической системы (генома) с одним из элементов объемной базы геномных сигнатур, а также рассмотрена их применимость для выявления мутирующих атак сигнатурными СОВ;

– предложен новый сигнатурный метод обнаружения мутирующих атак, основанный на механизме суффиксных деревьев, используемых при сборке и проверке схожести геномных последовательностей.

2. Исследование возможных решений. Требования, схожие с теми, что выдвигаются для сигнатурных СОВ относительно противодействия мутирующим атакам, предъявляются к алгоритмам биоинформатики, целью которых является сборка и сопоставление геномных последовательностей. Задачей обработки геномных последовательностей в биоинформатике является восстановление и упорядочивание больших цепочек ДНК длиной до миллиардов нуклеотидных кодов на основании информации, полученной в результате секвенирования [11-13]. Секвенирование – общее название биоинформационных методов, которые позволяют установить последовательность нуклеотидов в последовательности ДНК. В настоящее время нет ни одного метода секвенирования, который бы работал над геномными последовательностями целиком – все они устроены таким образом, что сначала готовится большое число нуклеотидных блоков (геном многократно клонируется и разрезается на блоки – риды), которые затем обрабатываются. Методы обработки ридов отличаются вариантами параллелизма и организации вычислений на структурах данных кодированного представления ридов.

Риды – последовательности, получаемые при секвенировании и содержащие информацию о фрагментах генома. Рид представляется строкой четырехбуквенного алфавита нуклеотидов, соответствующей фрагменту генома. Секвенаторы, в зависимости от механизма их работы, совершают ошибки, наиболее частая из которых – замена одного нуклеотида на другой (например, в риде находится нуклеотид Т, а в соответствующей позиции генома – А), а также ошибки пропуска ряда

нуклеотидов и вставки посторонних нуклеотидов в рид. При сборке генома из ридов необходимо устранить ошибки секвенирования. Для этого используются различные алгоритмы выравнивания последовательностей и поиска гомологически близких строк геномных кодов. Тем самым сборка и выравнивание геномов аналогичны задаче поиска подобных последовательностей среди сигнатур СОВ. Цель алгоритмов сборки генома, как и сигнатурных СОВ, состоит в нахождении совпадений рида (в СОВ – участка в последовательности команд системы) с одним из участков из объемной базы сигнатур. Алгоритмы биоинформатики полностью отвечают требованиям, предъявляемым к СОВ, устойчивым к мутирующим атакам, так как изменения, вносимые в атаки, проявляются в последовательности команд как вставка дополнительных элементов в последовательность и аналогичны ошибкам вставки лишних блоков нуклеотидов секвенатором. Качество и быстродействие биоинформационных алгоритмов обусловлено требуемой от них точностью и скоростью обработки больших последовательностей [14].

Классификация известных биоинформационных алгоритмов обработки геномных последовательностей представлена на рисунке 1.

Семейство алгоритмов De Novo [15] используется для сборки ранее неизвестного генома и основано на избыточности ридов, за счет которой восстанавливается порядок их следования. Алгоритмы Overlap Layout Concensus [16] и алгоритмы на графах де Брюина [17, 18] обладают квадратичной сложностью и поэтому не подходят для быстрой обработки больших последовательностей.

Алгоритмы выравнивания последовательностей на входе помимо набора ридов получают ранее восстановленный геном, который был собран до этого [19]. Оценивание схожести последовательностей, выполняемое при этом, упрощает процесс нахождения мутаций. Глобальное выравнивание [20] предполагает, что последовательности изначально гомологичны, и учитывает это сходство на протяжении всего выравнивания. Например, глобальное выравнивание Нидлмана-Вунша [21, 22] основано на оценке коллинеарности двух последовательностей и работает оптимально при сравнении очень схожих последовательностей. Соответствие выравненных символов задается матрицей схожести, значения ячеек которой растут при совпадении и уменьшаются при несовпадении элементов. Корректная работа алгоритма обусловлена свойством аддитивности, по которому делается оптимальный выбор на каждом шаге.

Локальное выравнивание ищет схожие блоки в последовательностях и выравнивает последовательности относительно блоков. Соответ-

ственно, для пары последовательностей может быть несколько локальных выравниваний. Алгоритм Смита-Уотермана [23, 24] учитывает локальные выравнивания схожих областей, которые попадают в разные наборы последовательностей. В отличие от схемы Нидлмана-Вунша значения в матрице схожести не могут опускаться ниже определенного минимума, и, таким образом, итоговое выравнивание разбивается на несколько оптимальных участков.

Множественное выравнивание трех и более геномных последовательностей предполагает, что входной набор последовательностей имеет эволюционную связь. Ввиду большей вычислительной сложности по сравнению с парным выравниванием, многие реализации множественного выравнивания используют эвристические алгоритмы.

Рис. 1. Классификация биоинформационных алгоритмов обработки последовательностей

Различают следующие алгоритмы, реализующие множественное выравнивание:

– прогрессивный алгоритм выравнивания [25, 26] реализует две стадии:

а) построение бинарного путеводного дерева, в котором листья являются последовательностями,

б) построение множественного выравнивания путем добавления последовательностей к растущему выравниванию согласно путеводному дереву.

Выравнивание может быть неудачным в случае набора сильно отдаленных друг от друга последовательностей. Ошибки, полученные на любой стадии растущего множественного выравнивания, доходят до результирующего выравнивания. Алгоритм требователен к схожести начальных последовательностей, что не подходит для решения поставленной задачи;

– итеративный алгоритм [25, 27] работает аналогично прогрессивному, но при этом он неоднократно перестраивает исходные выравнивания при добавлении новых последовательностей. Алгоритм может возвращаться к первоначально посчитанным парным выравниваниям и подвыравниваниям, содержащим подмножества последовательностей из запроса, и таким образом оптимизировать целевую функцию и повышать качество. Прогрессивное и итеративное выравнивания достаточно эффективны для одновременной обработки большого числа (100…1000) последовательностей, но, так как они эвристические, то они не гарантируют нахождения глобального оптимального выравнивания;

– скрытая марковская модель [12, 28] – вероятностная модель, которая может оценить правдоподобие для всех возможных комбинаций пропусков, совпадений или несовпадений для того, чтобы определить наиболее вероятное множественное выравнивание. Скрытая марковская модель может вычислять одно выравнивание с высоким весом, но также может сгенерировать семейство возможных выравниваний, которые затем могут быть оценены по их весу. Модель может быть использована для получения как глобальных, так и локальных выравниваний. Несмотря на то, что решение на базе скрытой марковской модели появилось сравнительно недавно, оно значительно улучшило вычислительную сложность, особенно для последовательностей, содержащих перекрывающиеся области.

Алгоритмы, основанные на скрытых марковских моделях, представляют множественное выравнивание в виде направленного ациклического графа, который состоит из серий узлов, представляющих собой возможные состояния в колонках выравнивания. В этом представлении абсолютно консервативная колонка (то есть последовательности во множественном выравнивании имеют в этой позиции определенный символ) кодируется как один узел со множеством исходящих соединений с символами, возможными в следующей позиции выравнивания. В терминах стандартной скрытой марковской модели наблюдаемые состояния – отдельные колонки выравнивания, а «скрытые» состояния представляют собой предполагаемую предковую последовательность, от которой последовательности из входного набора могли произойти. Аналогично прогрессивному выравниванию алгоритм требователен к

классификации сигнатур, но представляет более близкое к конкретной сигнатуре выравнивание.

Оптимизационные алгоритмы вычислительного интеллекта также используются для построения множественных выравниваний. Например, генетический алгоритм реализует гипотетическое эволюционное разделение серий возможных цепочек на фрагменты и повторную их перестройку с вводом разрывов в различные локации [29]. Алгоритм решает задачу поэтапного приближения к шаблону, что удовлетворяет цели сборки генома, но не позволяет быстро оценить схожесть пар последовательностей, что делает его непригодным для обнаружения вторжений.

Отдельно выделяют быстрые алгоритмы локального парного выравнивания – выравниватели коротких прочтений:

– алгоритм на базе хэш-таблиц, который использует хеш-функцию, трансформирующую строку в ключ быстрого поиска [30]. Наиболее простым способом было бы разбиение последовательности генома на слова, совпадающие по длине с ридом, но этот подход не работает, так как длинные слова обычно уникальны и их хранение требует слишком много места в памяти. Вместо этого используются хеширование более коротких блоков, которые встречаются гораздо чаще. После того, как с помощью хеш-функции получены подходящие позиции, можно картировать оставшуюся часть прочтения на геном. Подход разделения прочтения на несколько частей позволяет заложить в алгоритме возможность замен. Рид можно разбить на множество последовательностей со сдвигом в несколько нуклеотидов. Таким образом можно бороться с ошибками секвенирования (ошибочный фрагмент с обеих сторон окружен короткими последовательностями, которые в свою очередь успешно выравниваются).

Алгоритмы хеширования плохо справляются с повторами, так как сильно растет количество ридов, которое необходимо проверять. Для решения этой проблемы были разработаны суффиксные деревья [31, 32]. Преимущество суффиксных деревьев заключается в том, что повторы не увеличивают время работы этого алгоритма, так как повторяющиеся участки схлопываются в суффиксном дереве. Данный механизм работает крайне быстро при условии отсутствия ошибок и замен.

Сравнительная таблица 1 отражает основные свойства рассмотренных алгоритмов биоинформатики. Для дальнейшего исследования отобран механизм суффиксных деревьев, который дополнен предшествующим разбиением последовательностей на меньшие блоки для лучшей работы с мутирующими атаками. Такое решение избавляет от влияния мутационных вставок, пропусков и смены порядка команд, так как

в случае мутаций позволяет полностью восстановить из последовательности одну из наиболее близких сигнатур, находящихся в базе сигнатур.

Таблица 1 – Сравнение биоинформационных алгоритмов сборки и выравнивания геномов (m – длина анализируемого блока, n – длина генома, k – размер базы геномов, c – мощность алфавита геномов)

Алгоритм	Выравнивание мутированных последовательностей	Требовательность к чистым сигнатурам	Требовательность к большим базам сигнатур	Сложность построения структур данных	Сложность поиска	Объем памяти	Высокий коэффициент выравнивания
Алгоритм Ниддлмана-Вунша	–	+	+	$O(nmk)$	$O(nmk)$	$O(nmk)$	–
Алгоритм Смита-Ватермана	+	+	–	$O(nmk)$	$O(nmk)$	$O(nmk)$	–
Прогрессивный алгоритм	–	–	+	$O(n^2k^2)$	$O(m)$	$O(nk)$	–
Итеративный алгоритм	+	–	+	$O(n^2k^2)$	$O(m)$	$O(nk)$	+
Скрытая марковская модель	+	–	+	$O(nc^2)$	$O(m)$	$O(nk)$	+
Хэш-таблица	+	+	–	$O(nmk)$	$O(k)$	$O(nk)$	+
Суффиксное дерево	+	+	–	$O(nk)$	$O(mk)$	$O(nk)$	+

3. Метод обнаружения мутирующих атак на базе суффиксных деревьев. Бор – структура данных для хранения набора закодированных последовательностей, представляющая собой подвешенное дерево с символами на ребрах. Строки получаются последовательной записью всех символов, хранящихся на ребрах между корнем и терминальной вершиной. Размер бора линейно зависит от суммы длин всех строк. Поиск в бору занимает время, пропорциональное длине образца [33].

Рассмотрим бор, содержащий некоторый набор слов $s_1,...,s_k$. Количество вершин бора может достигать суммарной длины слов $|s_1|+...+|s_k|$. Для сокращения количества вершин рассмотрим такую цепочку вершин бора, что из каждой вершины исходит единственное ребро в следующую, и сожмем такую цепочку в одно ребро, а вместо буквы напишем на нем всю последовательность букв с ребер, которые

мы заменили. Эта последовательность букв является подстрокой некоторой строки s_i из набора, поэтому запишем на ребре только номер строки, а также начало и конец соответствующей подстроки. Сжатие всех строк в боре позволяет построить сжатый бор – корневое дерево, на каждом ребре которого написана непустая строка, обладающее следующими свойствами [33]:

– ни из какой вершины не выходят два ребра, строки для которых начинаются на один и тот же символ;

– если вершина не является корнем или листом дерева, из нее выходит не менее двух ребер.

Количество вершин в сжатом боре составляет $O(k)$, где k – количество строк в наборе. Суммарное количество вершин не превосходит $2k$. Сжатый бор занимает $O(k)$ памяти, однако для операций с ним необходимо явно хранить все строки s_i, поэтому по памяти аналогичный выигрыш не достигается.

Суффиксное дерево строки s – сжатый бор, построенный на всех суффиксах s. Такой бор занимает $O(|s|)$ памяти. Однако явное хранение всех суффиксов по отдельности не требуется: они все присутствуют в строке s. Это значит, что суффиксное дерево позволяет ответить на запросы «является ли строка t суффиксом s» и «является ли строка t префиксом суффикса, то есть подстрокой s» за время $O(|t|)$. Также оно позволяет получить следующую информацию о строке s и ее подстроках:

– количество различных подстрок строки s. Если спуститься в суффиксном дереве по пути, соответствующему подстроке, мы окажемся либо в вершине, либо посередине ребра (то есть будет пройдена только часть подстроки, соответствующей ребру). Количество различных подстрок s равно количеству различных позиций внутри суффиксного дерева, или сумме длин подстрок, написанных на ребрах, плюс один (положение в корне – пустая подстрока);

– длину наибольшего общего префикса для двух подстрок строки s. Общему префиксу двух строк соответствует общий участок двух путей, идущих от корня;

– лексикографический порядок суффиксов строки s. Обход суффиксного дерева, который в каждой вершине перебирает исходящие ребра в лексикографическом порядке первого символа ребра, перебирает

позиции в дереве в порядке возрастания строк. Отсюда следует, что порядок обхода листьев (позиций, соответствующих суффиксам) есть их лексикографическая сортировка.

Неявное суффиксное дерево строки s – дерево, полученное из суффиксного дерева $s\$$ удалением всех вхождений терминального символа $\$$ из меток ребер дерева, удалением после этого ребер без меток и затем удалением вершин, имеющих меньше двух потомков. Неявное суффиксное дерево префикса $s[1...i]$ строки s получается аналогично из суффиксного дерева для $s[1...i]\$$ удалением символов $\$$, дуг и вершин.

Неявное суффиксное дерево для любой строки s будет иметь меньше листьев, чем суффиксное дерево для строки $s\$$, в том и только том случае, если хотя бы один из суффиксов s является префиксом другого суффикса. Терминальный символ $\$$ добавлен к s как раз во избежание этой ситуации. Если s заканчивается символом, который больше нигде в s не появляется, то неявное суффиксное дерево для s будет иметь лист для каждого суффикса и, следовательно, будет настоящим суффиксным деревом.

Хотя неявное суффиксное дерево может иметь листья не для всех суффиксов, в нем закодированы все суффиксы s – каждый произносится символами какого-либо пути от корня этого неявного суффиксного дерева. Однако если этот путь не кончается листом, то не будет маркера, обозначающего конец пути. Таким образом, неявные суффиксные деревья сами по себе неинформативны.

Обобщенное суффиксное дерево набора строк $s_1 \dots s_n$ – суффиксное дерево, содержащее все суффиксы каждой из n строк. При построении такого дерева каждая строка должна дополняться уникальным символом маркера вне алфавита (или строкой), чтобы гарантировать, что суффикс не является подстрокой другого и представлен уникальным конечным узлом.

Для реализации поставленной задачи выявления вторжений на основе базы сигнатур создается обобщенное суффиксное дерево для сигнатур атак. В данном исследовании за основу взят алгоритм Укконена построения обобщенного суффиксного дерева и обеспечивающий приемлемую сложность [33]. Базовый алгоритм последовательно строит суффиксное дерево для всех префиксов исходного текста $S = s_1 s_2 \dots s_n$. На i-ом шаге неявное суффиксное дерево τ_{i-1} для префикса $s[1...i-1]$ достраивается до τ_i для префикса $s[1...i]$. Для этого

для каждого суффикса подстроки $s[1...i-1]$ выполняют спуск из корня дерева до конца суффикса и дописывают символ s_i. Алгоритм состоит из n этапов, на каждом их которых происходит продление всех суффиксов текущего префикса строки, что требует $O(n^2)$ времени. Общая асимптотика алгоритма – $O(n^3)$.

Пусть $x\alpha$ обозначает произвольную строку, где x – ее первый символ, а α – оставшаяся (возможно пустая) подстрока. Если для внутренней вершины v с путевой меткой $x\alpha$ существует другая вершина $s(v)$ с путевой меткой α, то ссылка из v в $s(v)$ называется суффиксной ссылкой. Для любой внутренней вершины v суффиксного дерева существует суффиксная ссылка, ведущая в некоторую внутреннюю вершину u (пример работы суффиксных ссылок представлен в [34]).

Рассмотрим применение суффиксных ссылок. Пусть только что был продлен суффикс $s[j...i-1]$ до суффикса $s[j...i]$. Теперь с помощью построенных суффиксных ссылок можно найти конец суффикса $s[j+1...i-1]$ в суффиксном дереве, чтобы продлить его до суффикса $s[j+1...i]$. Для этого проходят вверх по дереву до ближайшей внутренней вершины v, в которую ведет путь $s[j...r]$.

У вершины v всегда есть суффиксная ссылка, ведущая к вершине u, которой соответствует путь $s[j+1...r]$. Далее от вершины u следует пройти вниз по дереву к концу суффикса $s[j+1...i-1]$ и продлить его до суффикса $s[j+1...i]$. При этом подстрока $s[j+1...i-1]$ является суффиксом подстроки $s[j...i-1]$. Следовательно, после перехода по суффиксной ссылке в вершину, помеченную путевой меткой $s[j+1...r]$, можно дойти до места, которому соответствует метка $s[r+1...i-1]$, сравнивая не символы на ребрах, а лишь длину ребра по первому символу рассматриваемой части подстроки и длину самой этой подстроки.

В процессе построения суффиксного дерева уже построенные суффиксные ссылки никак не изменяются. Поэтому рассмотрим построение суффиксных ссылок для созданных вершин. Возьмем новую внутреннюю вершину v, которая была создана в результате продления суффикса $s[j...i-1]$. Вместо того, чтобы искать, куда должна указывать

суффиксная ссылка вершины v, проходя путь до корня дерева, продлим следующий суффикс $s[j+1...i-1]$. В этот момент можно проставить суффиксную ссылку для вершины v. Она будет указывать либо на существующую вершину, если следующий суффикс закончился в ней, либо на новую созданную. Таким образом, для вершины v точно найдется на следующем шаге алгоритма внутренняя вершина, в которую должна вести суффиксная ссылка.

Глубиной $d(v)$ вершины v является число ребер на пути от корня до вершины. При переходе по суффиксной ссылке глубина уменьшается не более чем на единицу. Число переходов по ребрам внутри фазы i – $O(i)$. В начале каждой фазы выполняется только один спуск от корня, а затем используются переходы по суффиксным ссылкам. Переходов внутри фазы алгоритма – $O(i)$. Фаза алгоритма состоит из i итераций, и кумулятивно получаем, что на одной итерации будет выполнено $O(1)$ действий. Асимптотика улучшенного алгоритма – $O(n^2)$.

Для дальнейшей оптимизации алгоритма до уровня $O(n)$ предлагается использовать линейное количество памяти. Метку каждого ребра будем сохранять как два числа – позиции ее самого левого и самого правого символов в исходном тексте. Если в какой-то момент работы алгоритма создан лист с меткой i (для суффикса, начинающегося в позиции i строки s), он останется листом во всех последовательных деревьях, созданных алгоритмом. Если правило продления применяется в продолжении суффикса, начинающего в позиции j, оно же и будет применяться во всех дальнейших продолжениях (от $j+1$ до i) до конца фазы алгоритма. Следовательно, в каждой фазе i алгоритм работает с суффиксами из диапазона $j...k, k \leqslant i$ вместо диапазона 1...i.

Алгоритм позволяет обнаруживать с помощью суффиксных деревьев исключительно те последовательности атак, точные копии которых занесены в базу сигнатур. Мутация анализируемой последовательности команд, составляющих атаку, приведут к ошибке второго рода. Данная проблема решена путем разбиения последовательностей на меньшие блоки, что избавляет от влияния мутационных вставок, пропусков и смены порядка команд, так как в случае мутаций позволяет восстанавливать из последовательности одну из наиболее близких сигнатур, находящихся в базе сигнатур.

Обозначим анализируемую последовательность команд $a = C_{a1}C_{a2}C_{a3}\ldots C_{an}$, а известные сигнатуры: $c_1 = C_{c_1,1}C_{c_1,2}C_{c_1,3}\ldots C_{c_1,n_1}$, $c_2 = C_{c_2,1}C_{c_2,2}C_{c_2,3}\ldots C_{c_2,n_2}$, $c_m = C_{c_m,1}C_{c_m,2}C_{c_m,3}\ldots C_{c_m,n_m}$. Каждый вектор атаки из известного множества атак разбивается на пересекающиеся блоки длиной k. На полученных участках строится суффиксное дерево. Для этого очередную сигнатуру $c_i = C_{c_i,1}C_{c_i,2}C_{c_i,3}\ldots C_{c_i,n_i}$ разделим на блоки длины k: $C_{c_i,1}C_{c_i,2}\ldots C_{c_i,k}$; $C_{c_i,2}C_{c_i,3}\ldots C_{c_i,k+1}$; $C_{c_i,n_i-k}C_{c_i,n_i-k+1}\ldots C_{c_i,n_i}$ и добавим полученные подпоследовательности в имеющееся суффиксное дерево.

При сравнении поступившей последовательности a с сигнатурами, по которым уже построено суффиксное дерево, ее аналогично разобьем на блоки длиной k, после чего проведем поиск каждого отдельного блока в суффиксном дереве и вычислим долю совпадений. Если эта доля превышает установленное пороговое значение, то последовательность a отнесем к атаке. Обучение итогового алгоритма сводится к подбору оптимального значения длины k и порога срабатывания.

Рассмотрим пример работы предложенного алгоритма. Пусть в базе сигнатур заданы следующие шаблоны атак:

(1) open, read, open, write, execute, connect, write, write, connect, execute;
(2) execute, execute, connect, open, write, execute, connect, open, read, write.

Построим суффиксное дерево на блоках, выделенных из заданных сигнатур при $k = 3$ (рис. 2, команды обозначены первым символом):

open, read, open; read, open, write; open, write, execute; write, execute, connect; execute, connect, write; connect, write, write; write, write, connect; write, connect, execute.

Разобьем на блоки вторую сигнатуру и добавим их в суффиксное дерево (рис. 3):

execute, execute, connect; execute, connect, open; connect, open, write; open, write, execute; write, execute, connect; execute, connect, open; connect, open, read; open, read, write.

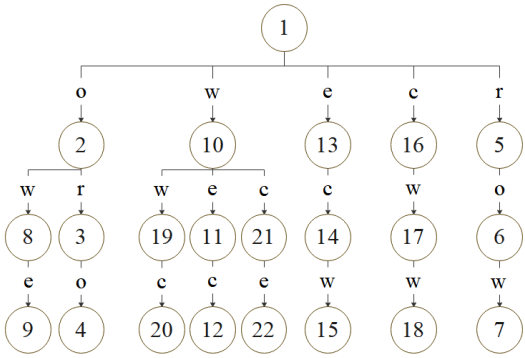

Рис. 2. Суффиксное дерево для первого вектора атаки

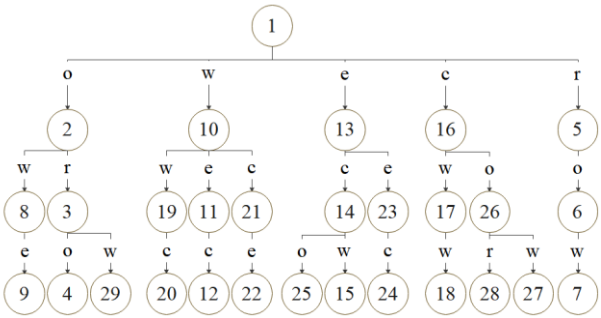

Рис. 3. Суффиксное дерево, дополненное вторым вектором атак

Проанализируем следующую поступившую последовательность на схожесть относительно базы сигнатур:

connect, open, write, execute, connect, write, open, write, execute, connect.

Разобьем данный вектор на блоки той же длины $k = 3$:

connect, open, write; open, write, execute; write, execute, connect; execute, connect, write; connect, write, open; write, open, write; open, write, execute; write, execute, connect.

Поиск в суффиксном дереве показывает, что 6 из 8 блоков (75%) содержатся в дереве. Участки connect, write, open и write, open, write в дереве не содержатся. При пороге 70% поступивший вектор определен как атака.

4. Экспериментальное исследование. Задача экспериментальной оценки созданного метода – оценка уровня ошибок и ресурсных затрат алгоритма при работе на тестовых наборах данных. Для исследований в сетевом режиме использован тестовый набор данных KDD Cup 1999 [35], представляющий собой размеченный датасет последовательностей сетевых пакетов. Использованы наборы для следующих сетевых атак: back dos; multihop r2l; satan probe; buffer_overflow u2r; neptune dos; smurf dos; ftp_write r2l; nmap probe; spy r2l; guess_passwd r2l; perl u2r; teardrop dos; imap r2l; phf r2l; warezclient r2l; ipsweep proe; pod dos; warezmaster r2l; land dos; portsweep probe; loadmodule u2r; rootkit u2r). Испытательный стенд развернут на платформе Intel Core i7, 32 Гб, SSD 1 Тб, операционная система MS Windows 10. Алгоритм выявления вторжений на базе суффиксных деревьев реализован на языке C.

График зависимости уровня ошибок при обнаружении атак от размера базы сигнатур представлен на рисунке 5 (приведен пример для параметров: размер блока $k = 4$, порог срабатывания – 80%.). С ростом размера базы сигнатур незначительно, в пределах десятых долей процента, увеличивается вероятность ошибки первого рода, не превышающая 1%. Вероятность ошибок второго рода при этом сокращается до уровня 0,08.

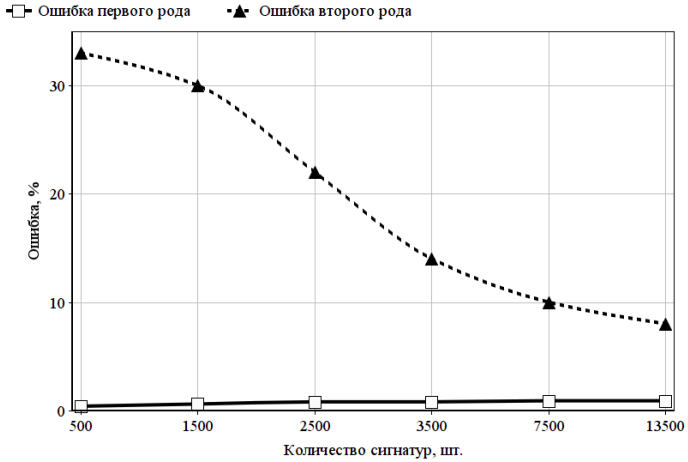

Рис. 5. Оценка уровня ошибок алгоритма ($k = 4$, порог – 80%)

Уровень ресурсозатрат суффиксного алгоритма в зависимости от размера базы сигнатур представлен на рисунках 6 и 7.

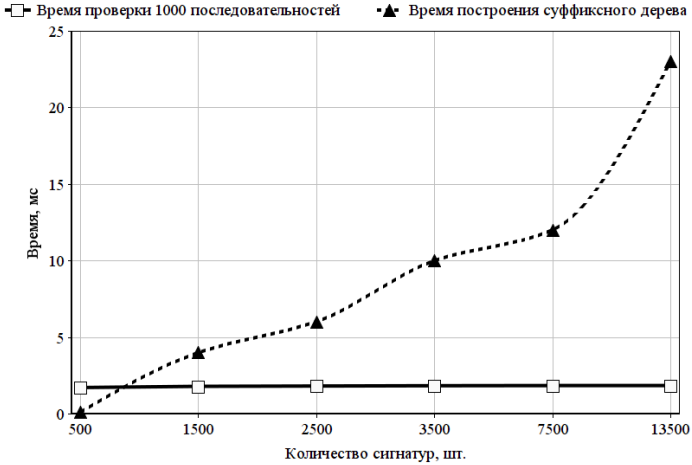

Рис. 6. Оценка временных затрат на проверку
и построение суффиксного дерева

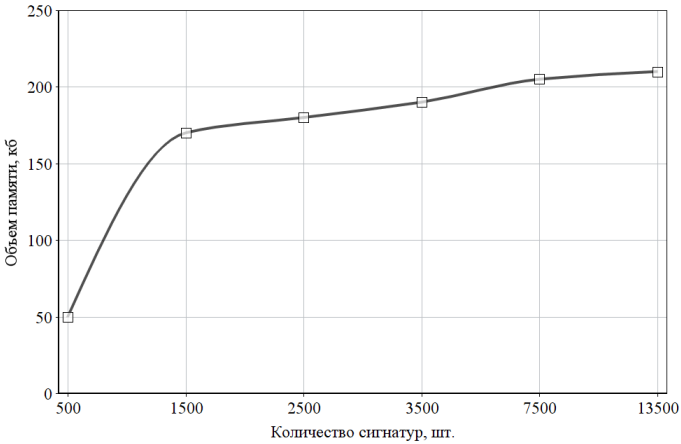

Рис. 7. Оценка объема памяти для хранения суффиксного дерева

Время, затрачиваемое на проверку последовательностей, и объем памяти, занимаемый суффиксным деревом, слабо меняются с наполнением базы сигнатур, что определяется ограниченной глубиной суффиксного дерева, в котором производится поиск, и детерминированностью переходов внутри дерева.

Время построения суффиксного дерева имеет почти линейную зависимость от размеров данных, на которых дерево строится, что подтверждает теоретические оценки алгоритма, приведенные ранее. Повторение одних и тех же элементов векторов и их блоков внутри базы сигнатур позволяет суффиксному дереву компактно хранить данные и эффективно использовать ресурс памяти.

Уровень ошибок второго рода в зависимости от доли мутируемой части анализируемых векторов приведен на рисунке 8 для базы сигнатур размером 13500 записей для тестового набора KDD Cup 1999 с искусственным внесением мутаций (диапазон доли мутаций – 0…30%).

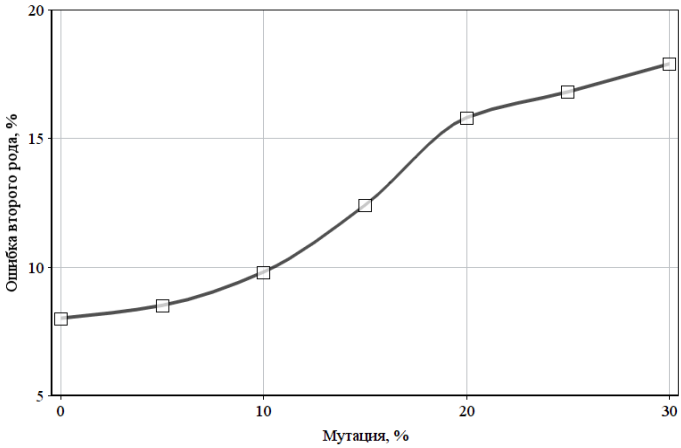

Рис. 8. Влияние мутаций на уровень ошибок алгоритма

Для дальнейшего улучшения алгоритма был проведен ряд модификаций, вследствие которых точность обнаружения атак увеличена до 95% при уровне мутаций 10%. Модифицированный алгоритм отличается по своей эффективности от исходного, комбинируя две системы деревьев, построенные на сигнатурах аномального и нормального поведения. Используется второе суффиксное дерево, построенное на известных шаблонах нормальных (не содержащих атаку) последовательностей. На этапе проверки, даже если проверяемая последовательность имеет высокие показатели совпадения по первому суффиксному дереву, построенному по сигнатурам атак, она не идентифицируется как атака до тех пор, пока коэффициент совпадения со вторым деревом не превысит определенный порог, заданный для второго дерева. Так, например, до модификации при $k = 3$, пороге для первого дерева 65%, пороге для

второго дерева 90% одиночное первое дерево является низкоэффективным, так как ввиду низкого порога система относит подавляющее число векторов к атакам (ошибки первого рода – около 90%), а при добавлении второго дерева ошибка снижается до 2%. При этом тандем суффиксных деревьев замедляет скорость работы алгоритма в N раз, где N – отношение количества векторов нормального поведения к количеству векторов атак в обучающей выборке, но из-за начальной линейной сложности данное замедление можно считать несущественным.

Для сравнения разработанного метода с традиционными СОВ использовались СОВ Suricata и Snort. В условиях отсутствия мутаций в сравниваемых последовательностях СОВ и разработанное решение демонстрируют идентичные показатели точности обнаружения атак. В случае распознавания мутирующих атак разработанное решение сохраняет устойчивость к изменениям в последовательностях. Суффиксный алгоритм позволяет пропускать несовпадающие элементы, и, следовательно, усовершенствовать сигнатурные СОВ, предоставив им новую возможность распознавать атаки, сигнатуры которых явно не содержатся в базе сигнатур. Помимо компактного использования ресурсов памяти для хранения больших баз сигнатур, метод обнаружения вторжений с помощью суффиксных деревьев обладает уникальным свойством – возможностью динамически расширять базу сигнатур во время работы СОВ. Сложность добавления новой ветви суффиксного дерева линейно зависит от длины добавляемого вектора, поэтому такое действие не сказывается на производительности и не требует перезаписи всей базы сигнатур. Это позволяет незамедлительно адаптировать СОВ к новым атакам, добавляемым к базе старых сигнатур.

5. Заключение. В исследовании проанализированы существующие решения в области биоинформатики – алгоритмы сборки и сопоставления геномных последовательностей. Рассмотрена их применимость для решения актуальной задачи идентификации мутирующих атак сигнатурными СОВ и выделен биоинформационный алгоритм обработки последовательностей на базе суффиксных деревьев, который позволяет добиться высокой точности обнаружения вторжений на уровне современных СОВ – более 90%, при этом превосходя их по экономичности использования оперативной памяти, быстродействию и устойчивости к возможным мутациям в векторах атак.

Для улучшения показателей точности проведен ряд модификаций разработанного алгоритма на базе суффиксных деревьев, вследствие которых точность обнаружения атак увеличена до 95% при уровне мутаций в последовательности до 10%.

Дальнейшие исследования направлены на адаптацию разработанного алгоритма для маломощных вычислительных устройств и реализацию для Интернета вещей сигнатурной СОВ на базе предложенного решения.

Разработанная технология обнаружения мутирующих атак с помощью суффиксных деревьев и СОВ, использующие его, могут применяться для обнаружения вторжений как в классических компьютерных сетях, так и в современных реконфигурируемых сетевых инфраструктурах с ограниченными ресурсами (Интернет вещей, сети киберфизических объектов, сенсорные сети).

Литература

1. *Khraisat A., Gondal I., Vamplew P., Kamruzzaman J.* Survey of intrusion detection systems: techniques, datasets and challenges // Cybersecurity. 2019. vol. 2. no. 1.
2. *Jatti S.A.V., Kishor Sontif V.J.K.* Intrusion detection systems // International Journal of Recent Technology and Engineering. 2019. vol. 8. no. 2. special issue 11. pp. 3976–3983.
3. *Branitskiy A.A., Kotenko I.V.* Analysis and classification of methods for network attack detection // SPIIRAS Proceedings. 2016. vol. 2. no. 45. pp. 207–244.
4. *Lakshminarayana D.H., Philips J., Tabrizi N.* A survey of intrusion detection techniques // In Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019. 2019. pp. 1122–1129.
5. *Platonov V.V., Semenov P.O.* An adaptive model of a distributed intrusion detection system // Automatic Control and Computer Sciences. 2017. vol. 51. no. 8. pp. 894–898.
6. *Platonov V.V., Semenov P.O.* Detection of Abnormal Traffic in Dynamic Computer Networks with Mobile Consumer Devices // Automatic Control and Computer Sciences, 2018. vol. 52. no. 8. pp. 959–964.
7. *Aljawarneh S.A., Moftah R.A., Maatuk A.M.* Investigations of automatic methods for detecting the polymorphic worms signatures // Future Generation Computer Systems. 2016. vol. 60. pp. 67–77.
8. *Khonde S.R., Venugopal U.* Hybrid architecture for distributed intrusion detection system // Ingenierie des Systemes d'Information. 2019. vol. 24. no. 1. pp. 19–28.
9. *Zhang W.A., Hong Z., Zhu J.W., Chen B.* A survey of network intrusion detection methods for industrial control systems // Kongzhi yu Juece/Control and Decision. 2019. vol. 34. no. 11. pp. 2277–2288.
10. *Seoane Fernández J.A., Miguélez Rico M.* Bio-Inspired Algorithms in Bioinformatics I // Encyclopedia of Artificial Intelligence. 2011.
11. *Levshun D, Gaifulina D., Chechulin A., Kotenko I.* Problematic issues of information security of cyber-physical systems // SPIIRAS Proceedings. 2020. vol. 19. no. 5. pp. 1050–1088.
12. *Coull S., Branch J., Szymanski B., Breimer E.* Intrusion detection: A bioinformatics approach // In Proceedings Annual Computer Security Applications Conference, ACSAC. 2003. vol. 2003-January. pp. 24–33.
13. *Lavrova D., Zaitceva E., Zegzhda P.* Bio-inspired approach to self-regulation for industrial dynamic network infrastructure // CEUR Workshop Proceedings. 2019. vol. 2603. pp. 34–39.
14. *Miller W.* An Introduction to Bioinformatics Algorithms // Journal of the American Statistical Association. 2006. vol. 101. no. 474. pp. 855–855.

15. *Sohn J. Il, Nam J.W.* The present and future of de novo whole-genome assembly // Briefings in Bioinformatics. 2018. vol. 19, no. 1, pp. 23–40.
16. *Recanati A., Brüls T., D'Aspremont A.* A spectral algorithm for fast de novo layout of uncorrected long nanopore reads // Bioinformatics. 2017. vol. 33, no. 20. pp. 3188–3194.
17. *Rizzi R., et al.* Overlap graphs and de Bruijn graphs: data structures for de novo genome assembly in the big data era // Quantitative Biology. 2019. vol. 7, no. 4. pp. 278–292.
18. *Wittler R.* Alignment- And reference-free phylogenomics with colored de Bruijn graphs // Algorithms for Molecular Biology. 2020. vol. 15. no. 1.
19. *Tan T.W., Lee E.* Sequence Alignment // Beginners Guide to Bioinformatics for High Throughput Sequencing. 2018. pp. 81–115.
20. *Muhamad F.N., Ahmad R.B., Asi S.M., Murad M.N.* Performance Analysis of Needleman-Wunsch Algorithm (Global) and Smith-Waterman Algorithm (Local) in Reducing Search Space and Time for DNA Sequence Alignment // Journal of Physics: Conference Series. 2018. vol. 1019. no. 1.
21. *Lee Y.S., Kim Y.S., Uy R.L.* Serial and parallel implementation of Needleman-Wunsch algorithm // International Journal of Advances in Intelligent Informatics. 2020. vol. 6. no. 1. pp. 97–108.
22. *Čavojský M., Drozda M., Balogh Z.* Analysis and experimental evaluation of the Needleman-Wunsch algorithm for trajectory comparison // Expert Systems with Applications. 2021. vol. 165.
23. *Sun J., Chen K., Hao Z.* Pairwise alignment for very long nucleic acid sequences // Biochemical and Biophysical Research Communications. 2018. vol. 502. no. 3. pp. 313–317.
24. *Zou H., Tang S., Yu C., Fu H., Li Y., Tang W.* ASW: Accelerating Smith–Waterman Algorithm on Coupled CPU-GPU Architecture // International Journal of Parallel Programming. 2019. vol. 47. no. 3. pp. 388–402.
25. *Chowdhury B., Garai G.* A review on multiple sequence alignment from the perspective of genetic algorithm // Genomics. 2017. vol. 109. no. 5–6. pp. 419–431.
26. *Dijkstra M.J.J., Van Der Ploeg A.J., Feenstra K. A., Fokkink W.J., Abeln S., Heringa J.* Tailor-made multiple sequence alignments using the PRALINE 2 alignment toolkit // Bioinformatics. 2019. vol. 35. no. 24. pp. 5315–5317.
27. *Chen S., Yang S., Zhou M., Burd R., Marsic I.* Process-Oriented Iterative Multiple Alignment for Medical Process Mining // In IEEE International Conference on Data Mining Workshops, ICDMW. 2017. vol. 2017-November. pp. 438–445.
28. *Ye N.* Markov Chain Models and Hidden Markov Models // Data Mining. 2021. pp. 287–305.
29. *Behera N., Jeevitesh M.S., Jose J., Kant K., Dey A., Mazher J.* Higher accuracy protein multiple sequence alignments by genetic algorithm // Procedia Computer Science. 2017. vol. 108. pp. 1135–1144.
30. *Cui X., Shi H., Zhao J., Ge Y., Yin Y., Zhao K.* High Accuracy Short Reads Alignment Using Multiple Hash Index Tables on FPGA Platform // In Proceedings of 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference, ITOEC. 2020. pp. 567–573.
31. *Marçais G., Delcher A.L., Phillippy A.M., Coston R., Salzberg S.L., Zimin A.* MUMmer4: A fast and versatile genome alignment system // PLoS Computational Biology. 2018. vol. 14. no. 1. 2018.
32. *Kay M.* Substring alignment using suffix trees // Lecture Notes in Computer Science. 2004. vol. 2945. pp. 275–282.
33. *Ukkonen E.* On-line construction of suffix trees // Algorithmica. 1995. vol. 14. no. 3. pp. 249–260.

34. *Breslauer D., Italiano G.F.* On suffix extensions in suffix trees // Theoretical Computer Science. 2012. vol. 457. pp. 27–34.
35. KDD Cup 1999 Data: URL: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата доступа: 10.04.2021).

Зегжда Дмитрий Петрович — д-р техн. наук, профессор РАН, директор института, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: моделирование безопасности, методы киберустойчивости, технология безопасности систем больших данных. Число научных публикаций — 340. dmitry@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Калинин Максим Олегович — д-р техн. наук, профессор, профессор, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: анализ киберрисков, методы машинного обучения, методы оценки безопасности реконфигурируемых систем на моделях. Число научных публикаций — 320. max@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Крундышев Василий Михайлович — аспирант, ассистент, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: методы искусственного интеллекта в обработке данных о защищенности сложных систем, технологии виртуализации, методы моделирования средств защиты. Число научных публикаций — 60. vmk@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Лаврова Дарья Сергеевна — д-р техн. наук, профессор, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: биоподобные методы кибербезопасности, методы высокопроизводительного анализа больших данных, технологии обработки событий безопасности. Число научных публикаций — 270. lavrova@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Москвин Дмитрий Андреевич — к-т техн. наук, доцент, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: верификация безопасности, методы model checking, эвристические и поведенческие модели безопасности. Число научных публикаций — 120. moskvin@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Павленко Евгений Юрьевич — к-т техн. наук, доцент, Институт кибербезопасности и защиты информации, Санкт-Петербургский политехнический университет Петра Великого. Область научных интересов: адаптивные методы управления безопасностью, модели управления и принятия решений, графовые алгоритмы моделирования. Число научных публикаций — 190. pavlenko@ibks.spbstu.ru; Политехническая улица, д. 29, г. Санкт-Петербург, 192251, РФ; р.т.: +7(812)5527632, факс: +7(812)5527632.

Поддержка исследований. Работа выполнена в рамках Государственного задания на проведение фундаментальных исследований (код темы 0784-2020-0026).

DOI 10.15622/ia.20.4.3

D. ZEGZHDA, M. KALININ, V. KRUNDYSHEV, D. LAVROVA,
D. MOSKVIN, E. PAVLENKO

APPLICATION OF BIOINFORMATICS ALGORITHMS FOR POLYMORPHIC CYBERATTACKS DETECTION

Zegzhda D., Kalinin M., Kundyshev V., Lavrova D., Moskvin D., Pavlenko E. **Application of Bioinformatics Algorithms for Polymorphic Cyberattacks Detection.**

Abstract. The functionality of any system can be represented as a set of commands that lead to a change in the state of the system. The intrusion detection problem for signature-based intrusion detection systems is equivalent to matching the sequences of operational commands executed by the protected system to known attack signatures. Various mutations in attack vectors (including replacing commands with equivalent ones, rearranging the commands and their blocks, adding garbage and empty commands into the sequence) reduce the effectiveness and accuracy of the intrusion detection. The article analyzes the existing solutions in the field of bioinformatics and considers their applicability for solving the problem of identifying polymorphic attacks by signature-based intrusion detection systems. A new approach to the detection of polymorphic attacks based on the suffix tree technology applied in the assembly and verification of the similarity of genomic sequences is discussed. The use of bioinformatics technology allows us to achieve high accuracy of intrusion detection at the level of modern intrusion detection systems (more than 0.90), while surpassing them in terms of cost-effectiveness of storage resources, speed and readiness to changes in attack vectors. To improve the accuracy indicators, a number of modifications of the developed algorithm have been carried out, as a result of which the accuracy of detecting attacks increased by up to 0.95 with the level of mutations in the sequence up to 10%. The developed approach can be used for intrusion detection both in conventional computer networks and in modern reconfigurable network infrastructures with limited resources (Internet of Things, networks of cyber-physical objects, wireless sensor networks).

Keywords: Ukkonen Algorithm, Security, Bioinformatics, Alignment, Mutation, Intrusion Detection, Polymorphism, Signature, Suffix Tree.

Zegzhda Dmitry — Dr.Sc., Professor of the Russian Academy of Sciences, Director of Institute, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: security modeling, cyber resilience methods, big data system security technology. The number of publications – 340. dmitry@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Kalinin Maxim — Dr.Sc., Professor, Professor, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: cyber risk analysis, machine learning methods, methods for evaluating the security of reconfigurable systems on models. The number of scientific publications – 320. max@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Krundyshev Vasiliy — Ph.D student, Junior Researcher, Assistant, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: methods of artificial intelligence in the processing of data on the security of complex

systems, virtualization technologies, methods of modeling security tools. The number of scientific publications – 60. vmk@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Lavrova Daria — Dr.Sc., Professor, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: bioinspired methods for cybersecurity, methods of high-performance analysis of big data, security event processing technologies. The number of scientific publications – 270. lavrova@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Moskvin Dmitry — PhD, Associate Professor, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: security verification, model checking methods, heuristic and behavioral security models. The number of scientific publications – 120. moskvin@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Pavlenko Evgeny — PhD, Associate Professor, Institute for Cybersecurity and Information Protection, Peter the Great St. Petersburg Polytechnic University. Research interests: adaptive methods of security management, models of management and decision making, graph modeling algorithms. The number of scientific publications – 190. pavlenko@ibks.spbstu.ru; 29, Politekhnicheskaya ul., St. Petersburg, 192251, Russian Federation; office phone: +7(812)5527632, fax: +7(812)5527632.

Acknowledgements. The work was performed as part of the State assignment for basic research (topic code 0784-2020-0026).

References

1. Khraisat A., Gondal I., Vamplew P., Kamruzzaman J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*. 2019. vol. 2. no. 1.
2. Jatti S.A.V., Kishor Sontif V.J.K. Intrusion detection systems. *International Journal of Recent Technology and Engineering*. 2019. vol. 8. no. 2. special issue 11. pp. 3976–3983.
3. Branitskiy A.A., Kotenko I.V. Analysis and classification of methods for network attack detection. *SPIIRAS Proceedings*. 2016. vol. 2. no. 45. pp. 207–244.
4. Lakshminarayana D.H., Philips J., Tabrizi N. A survey of intrusion detection techniques. *In Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*. 2019. pp. 1122–1129.
5. Platonov V.V., Semenov P.O. An adaptive model of a distributed intrusion detection system. *Automatic Control and Computer Sciences*. 2017. vol. 51. no. 8. pp. 894–898.
6. Platonov V.V., Semenov P.O. Detection of Abnormal Traffic in Dynamic Computer Networks with Mobile Consumer Devices. *Automatic Control and Computer Sciences*, 2018. vol. 52. no. 8. pp. 959–964.
7. Aljawarneh S.A., Moftah R.A., Maatuk A.M. Investigations of automatic methods for detecting the polymorphic worms signatures. *Future Generation Computer Systems*. 2016. vol. 60. pp. 67–77.
8. Khonde S.R., Venugopal U. Hybrid architecture for distributed intrusion detection system. *Ingenierie des Systemes d'Information*. 2019. vol. 24. no. 1. pp. 19–28.
9. Zhang W.A., Hong Z., Zhu J.W., Chen B. A survey of network intrusion detection methods for industrial control systems. *Kongzhi yu Juece/Control and Decision*. 2019. vol. 34. no. 11. pp. 2277–2288.

10. Levshun D, Gaifulina D., Chechulin A., Kotenko I. Problematic issues of information security of cyber-physical systems. *SPIIRAS Proceedings*. 2020. vol. 19. no. 5. pp. 1050–1088.
11. Seoane Fernández J.A., Miguélez Rico M. Bio-Inspired Algorithms in Bioinformatics I. *Encyclopedia of Artificial Intelligence*, 2011.
12. Coull S., Branch J., Szymanski B., Breimer E. Intrusion detection: A bioinformatics approach. In Proceedings Annual Computer Security Applications Conference, ACSAC. 2003. vol. 2003-January. pp. 24–33.
13. Lavrova D., Zaitceva E., Zegzhda P. Bio-inspired approach to self-regulation for industrial dynamic network infrastructure. CEUR Workshop Proceedings. 2019. vol. 2603. pp. 34–39.
14. Miller W. An Introduction to Bioinformatics Algorithms. *Journal of the American Statistical Association*. 2006. vol. 101. no. 474. pp. 855–855.
15. Sohn J. Il, Nam J.W. The present and future of de novo whole-genome assembly. *Briefings in Bioinformatics*. 2018. vol. 19. no. 1. pp. 23–40.
16. Recanati A., Brüls T., D'Aspremont A. A spectral algorithm for fast de novo layout of uncorrected long nanopore reads. *Bioinformatics.* 2017. vol. 33, no. 20. pp. 3188–3194.
17. Rizzi R., et al. Overlap graphs and de Bruijn graphs: data structures for de novo genome assembly in the big data era. *Quantitative Biology*. 2019. vol. 7. no. 4. pp. 278–292.
18. Wittler R. Alignment- And reference-free phylogenomics with colored de Bruijn graphs. *Algorithms for Molecular Biology*. 2020. vol. 15. no. 1.
19. Tan T.W., Lee E. Sequence Alignment. *In Beginners Guide to Bioinformatics for High Throughput Sequencing*. 2018. pp. 81–115.
20. Muhamad F.N., Ahmad R.B., Asi S.M., Murad M.N. Performance Analysis of Needleman-Wunsch Algorithm (Global) and Smith-Waterman Algorithm (Local) in Reducing Search Space and Time for DNA Sequence Alignment. *Journal of Physics: Conference Series.* 2018. vol. 1019. no. 1.
21. Lee Y.S., Kim Y.S., Uy R.L. Serial and parallel implementation of Needleman-Wunsch algorithm. *International Journal of Advances in Intelligent Informatics*. 2020. vol. 6. no. 1. pp. 97–108.
22. Čavojský M., Drozda M., Balogh Z. Analysis and experimental evaluation of the Needleman-Wunsch algorithm for trajectory comparison. *Expert Systems with Applications*. 2021. vol. 165.
23. Sun J., Chen K., Hao Z. Pairwise alignment for very long nucleic acid sequences. *Biochemical and Biophysical Research Communications*. 2018. vol. 502. no. 3. pp. 313–317.
24. Zou H., Tang S., Yu C., Fu H., Li Y., Tang W. ASW: Accelerating Smith–Waterman Algorithm on Coupled CPU-GPU Architecture. *International Journal of Parallel Programming*. 2019. vol. 47. no. 3. pp. 388–402.
25. Chowdhury B., Garai G. A review on multiple sequence alignment from the perspective of genetic algorithm. *Genomics.* 2017. vol. 109. no. 5–6. pp. 419–431.
26. Dijkstra M.J.J., Van Der Ploeg A.J., Feenstra K. A., Fokkink W.J., Abeln S., Heringa J. Tailor-made multiple sequence alignments using the PRALINE 2 alignment toolkit. *Bioinformatics.* 2019. vol. 35. no. 24. pp. 5315–5317.
27. Chen S., Yang S., Zhou M., Burd R., Marsic I. Process-Oriented Iterative Multiple Alignment for Medical Process Mining. In IEEE International Conference on Data Mining Workshops, ICDMW. 2017. vol. 2017-November. pp. 438–445.
28. Ye N. Markov Chain Models and Hidden Markov Models. *Data Mining.* 2021. pp. 287–305.
29. Behera N., Jeevitesh M.S., Jose J., Kant K., Dey A., Mazher J. Higher accuracy protein multiple sequence alignments by genetic algorithm. *Procedia Computer Science.* 2017. vol. 108. pp. 1135–1144.

30. Cui X., Shi H., Zhao J., Ge Y., Yin Y., Zhao K. High Accuracy Short Reads Alignment Using Multiple Hash Index Tables on FPGA Platform. In Proceedings of IEEE 5th Information Technology and Mechatronics Engineering Conference, ITOEC. 2020. pp. 567–573.
31. Marçais G., Delcher A.L., Phillippy A.M., Coston R., Salzberg S.L., Zimin A. MUMmer4: A fast and versatile genome alignment system. *PLoS Computational Biology*. 2018. vol. 14. no. 1.
32. Kay M. Substring alignment using suffix trees. *Lecture Notes in Computer Science*. 2004. vol. 2945. pp. 275–282.
33. Ukkonen E. On-line construction of suffix trees. *Algorithmica*. 1995. vol. 14. no. 3. pp. 249–260.
34. Breslauer D., Italiano G.F. On suffix extensions in suffix trees. Theoretical Computer Science. 2012. vol. 457, pp. 27–34.
35. KDD Cup 1999 Data. Available at: kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed: 10.04.2021).