УДК 004.62

МЕТОД ФОРМИРОВАНИЯ МНОГОУРОВНЕВЫХ ПОСЛЕДОВАТЕЛЬНЫХ ПАТТЕРНОВ

А.В. Молдавская

Дослідження присвячене проблемі великого обсягу результатів, що отримують у процесі секвенційного аналізу послідовних даних. Запропоновано новий різновид послідовних патернов – багатовимірні послідовні патерни, описано метод їх отримання. Висунуто функціональні вимоги до програмної реалізації запропонованого методу. Продемонстровано результати експериментів, проведених на реальних даних про поведінку зловмисних програм.
Ключові слова: інтелектуальний аналіз даних, секвенційний аналіз, регулярні вирази.

Исследование посвящено проблеме большого объёма результатов, получаемых в процессе секвенциального анализа последовательных данных. Предложена новая разновидность последовательных паттернов – многомерные последовательные паттерны, описан метод их получения. Выдвинуты функциональные требования к программной реализации предложенного метода. Продемонстрированы результаты экспериментов, проведенных на реальных данных о поведении вредоносных программ.
Ключевые слова: интеллектуальный анализ данных, секвенциальный анализ, регулярные выражения.

The research is dedicated to the problem of large volumes of results acquired from sequential pattern mining. The new form of sequential patterns is proposed. The requirements for a programmed implementation of the described method are introduced. The results of experiments based on real malware behavior data are demonstrated.
Key words: data mining, sequential pattern mining, regular expressions.

Введение

Одной из важных задач анализа данных является выделение закономерностей. Для последовательных данных эту задачу решает в частности такая область интеллектуального анализа данных (data mining) как секвенциальный анализ или анализ последовательных паттернов (sequential pattern mining). Он позволяет выявлять часто встречающиеся участки в строках и в последовательностях наборов элементов. В качестве примеров применения можно привести анализ поведения покупателя, анализ поведения вредоносных программ, исследование свойств белковых цепочек. Одной из самых значимых проблем этой области является объёмность получаемых результатов, затрудняющая прочтение и понимание выделенных знаний пользователем [1–4].

Целью исследования является создание разновидности паттерна, которая была бы более ёмкой и удобной для восприятия, чем существующие. Для этого решаются следующие задачи:

– рассмотрение существующих методов секвенциального анализа и соответствующих этим методам видов паттернов;

– разработка нового вида паттерна – многоуровневого паттерна;

– разработка метода формирования многоуровневых паттернов, которые представляли бы полученные знания от общего к частному;

– выделение особенностей практической реализации данного метода;

– апробация на экспериментах, с выбором конкретных алгоритмов секвенциального анализа и применением их на разных этапах метода.

1. Секвенциальный анализ

Секвенциальный анализ (sequential pattern mining, поиск/добыча последовательных шаблонов) – это разновидность интеллектуального анализа данных (data mining). Объектом секвенциального анализа является база последовательностей.

Последовательность – это кортеж из наборов элементов (itemsets) - непустых множеств одновременно встречающихся элементов [1]. Пример последовательности: $S=<\{a\},\{a,b,c\},\{b\},\{b,c\},\{a,d\}>$. Последовательности представляют собой наборы одновременно встречающихся элементов, записанные в порядке их возникновения при наблюдении. На практике это используется, например, для отображения товаров, приобретённых пользователем одновременно, т. е. в рамках одной покупки. Тогда один набор элементов будет соответствовать содержимому одной покупки, а вся последовательность будет представлять собой череду покупок, сделанных пользователем за всё время наблюдения.

В случае, если необходимо проанализировать набор строк, каждый символ строки считается единичным набором элементов, а вся строка – последовательностью [3]. Пример: $S=<\{a\},\{b\},\{b\},\{c\},\{a\},\{d\}>$.

Будем называть такие единичные наборы элементов просто элементами последовательности, в отличие от элементов набора. Пример последовательности с указанием её составных частей показан на рис. 1.

Целью секвенциального анализа является получение часто встречающихся подпоследовательностей, которые называются последовательными шаблонами, или последовательными паттернами [2].

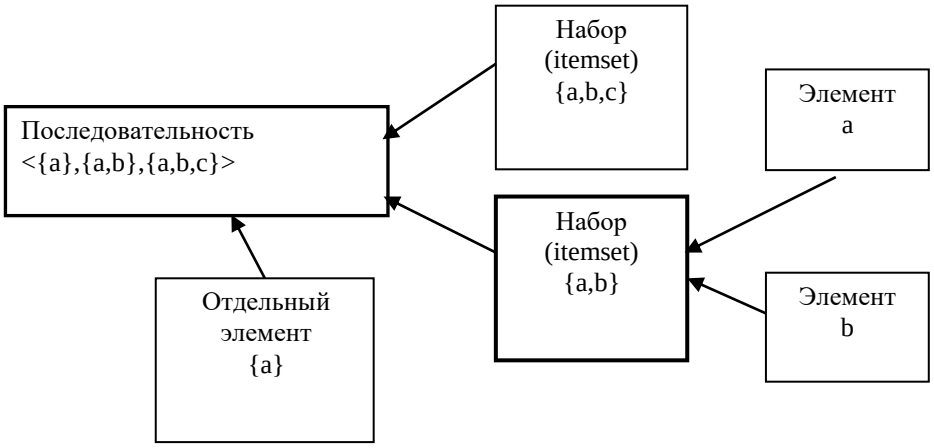

Рис. 1. Последовательность и её составляющие

Последовательным паттерном (образцом, шаблоном, англ. sequential pattern) называется последовательность элементов, являющаяся часто встречающейся подпоследовательностью некоторых последовательностей заданной базы. Подпоследовательность считается часто встречающейся, если её можно выделить из не менее чем s исходных последовательностей. Величина s называется поддержкой. Он характеризует количество последовательностей из базы, в которые входит подпоследовательность и обычно задаётся в процентах.

Одной из главных проблем секвенциального анализа является сокращение объёмов результирующей выборки, поскольку большие объёмы усложняют интерпретацию результатов [2]. Поэтому на практике проводят поиск паттернов по уточнённым определениям. К примеру, последовательный паттерн P_a называется закрытым, если не существует такого последовательного паттерна P_b, которая при такой же поддержке был бы надпоследовательностью для P_a [4]. Соответственно, задача добычи закрытых последовательных паттернов может быть определена следующим образом: выделить из базы последовательностей все паттерны, являющиеся закрытыми.

Формой закрытого последовательного паттерна является максимальный последовательный паттерн. Максимальным последовательным паттерном является такой закрытый паттерн, который не входит ни в один другой закрытый паттерн. Понятие поддержки, входившее в определение закрытого последовательного паттерна, не учитывается при определении максимального последовательного паттерна [4].

Все эти паттерны являются также наборами последовательностей. Логично предположить, что их можно анализировать так же, как исходные последовательности, получая более общие закономерности.

2. Многоуровневые паттерны. Метод составления многоуровневых паттернов средствами data mining

Назовём многоуровневым такой паттерн, элементами которого являются другие паттерны, полученные в результате секвенциального анализа из исходной выборки последовательностей. Очевидно, что эта исходная выборка также может состоять из паттернов, полученных на ещё более раннем этапе. Пример такого паттерна показан на рис. 2. Этот паттерн состоит из трёх элементов: $P = <\{A\}, \{B\}, \{C\}>$, при этом каждый элемент сам является паттерном и состоит из некоторых элементов. Например, $P_A = <\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}>$.

Опишем разработанный нами метод составления многоуровневых патттернов на основе секвенциального анализа на примере анализа строк, т. е. последовательностей, состоящих из единичных наборов элементов.

Пусть имеется выборка последовательностей D_s^0 из m последовательностей элементов. В результате работы алгоритма секвенциального анализа на основе выборки D_s^0 будет получена выборка паттернов D_p^1 объёма n. Конкретная величина объём выборки паттернов зависит от выбранного алгоритма и его настроек. Назовём паттерны, полученные в данной выборке, паттернами 1-го уровня. Уровнем 0 будем считать отдельные символы.

Составим для каждого найденного паттерна регулярное выражение вида:

$$a_1^i \mid a_2^i \mid \ldots \mid a_k^i, \qquad (1)$$

где $a_1^i \ldots \ldots a_k^i$ – элементы, принадлежащие *i*-му найденному паттерну длины k.

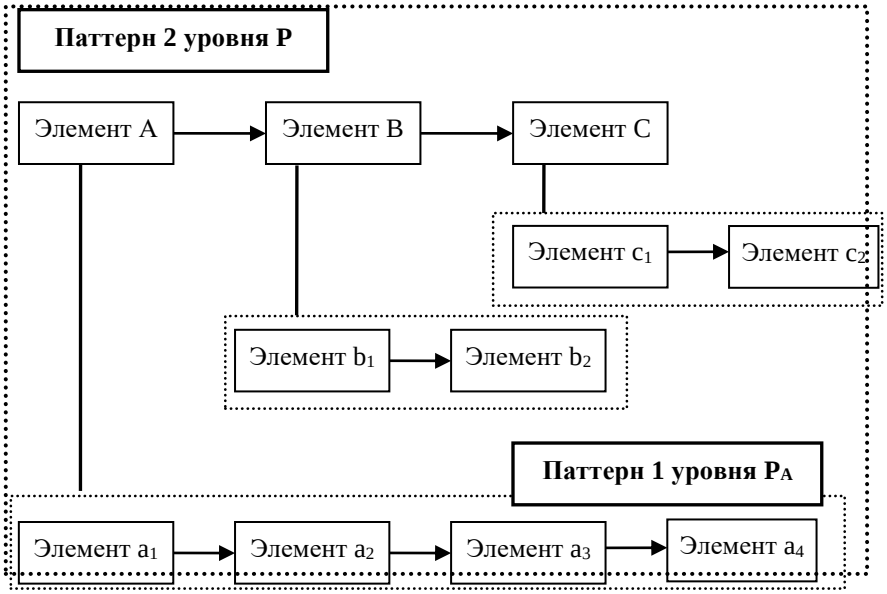

Рис. 2. Пример многоуровневого паттерна

Пропустим выборку последовательность D_s^0 через регулярное выражение (1). Это позволит получить выборку последовательностей D_s^1 объёма m_1, которую составляют упорядоченные последовательности паттернов, содержащих, в свою очередь, последовательности элементов.

Получим теперь следующий уровень многоуровневого паттерна, где элементам будут соответствовать паттерны, полученные из выборки D_s^1. После работы алгоритма секвенциального анализа будет получена выборка паттернов D_p^2 объёма n_2. Их элементами будут паттерны 1-го уровня, так как именно с их помощью представлены поведения в выборке D_b^1. Назовём эти паттерны паттернами 2-го уровня.

Следует заметить, что алгоритм не обязательно должен быть тем же, что и при получении выборки D_p^1. Работа алгоритма и получаемые результаты зависят от длины обрабатываемых последовательностей и их количества, а в выборке D_s^1 эти величины будут меньше, чем в D_s^0.

Для следующей итерации, как и ранее для D_p^1, восстановим последовательность расположения паттернов 1-го уровня, чтобы получить выборку последовательностей. Составим для каждого паттерна 2-го уровня регулярное выражение вида

$$p_1 \mid p_2 \mid \ldots \mid p_{n2}, \tag{2}$$

где $p_1 \ldots p_{n2}$ – паттерны найденной выборки D_p^2.

Процесс поиска паттернов останавливается в случае, если на очередном шаге не были выявлены новые паттерны или если было достигнуто необходимое количество уровней.

3. Функциональные требования к программной системе построения многоуровневых паттернов

При реализации программной системы, которая могла бы строить многоуровневые паттерны на основе методов секвенциального анализа, не обходимо выполнить следующие требования:

1. Система должна получать данные в виде последовательностей и преобразовывать их в форму, пригодную для обработки алгоритмами секвенциального анализа.

2. Система должна иметь возможность для задания пользователем параметров анализа и выбора конкретных алгоритмов на каждом этапе работы метода построения многоуровневых паттернов. Сами алгоритмы могут быть как реализованы в самой системе, так и подключены из внешних библиотек.

3. Система должна сохранять паттерны, полученные на каждом этапе работы. Также она должна кодировать их таким образом, чтобы они представляли собой элементы для работы следующих этапов.

4. Система должна предоставлять интерфейс для поуровневого просмотра результирующих паттернов.

Остановимся подробнее на п. 3. Возьмём для примера программную библиотеку алгоритмов секвенциального анализа, описанную в [5]. Для своей работы она требует кодирования элементов в виде чисел и разделения наборов элементов друг от друга разделителем «-1». Поэтому для применения данного метода с этой библиотекой потребуется:

1) каждому возможному элементу присвоить числовой код и сохранить соответствие кода и элемента;

2) на каждом этапе работы метода каждому найденному паттерну присвоить числовой код, не пересекающийся с кодами из п. 1 и сохранить соответствие кода и паттерна.

Практическая реализация этих требований возможна при использовании реляционной базы данных.

4. Результаты экспериментов

Продемонстрируем результаты экспериментов по секвенциальному анализу поведения вредоносных программ с последующим построением многоуровневых паттернов. Поясним выбор исходных данных. Вредоносные программы демонстрируют типовые последовательности действий, однако часто пытаются маскировать их незначительными действиями [6]. Поведение вредоносной программы можно представить в виде строки, состоящей из последовательности WinAPI-функций, вызываемых ею в процессе работы. Каждую WinAPI-функцию будем считать единичным набором элементов, как это описано в предыдущей главе для строк.

Для экспериментов использовалась коллекция вредоносных программ, описанная в [7], содержащая 3157 отчётов о поведении вредоносных программ в формате XML. Отчёты представлены в виде последовательностей WinAPI. Данная коллекция охватывает фазы жизненного цикла вредоносных программ, которые не требуют сетевого взаимодействия. Из выборки были исключены отчёты по вредоносным программам неопределённых семейств, а также отчёты о семействах с количеством экземпляров вредоносов меньше трёх. Backdoor – 595, Virus – 94, Worm – 224, P2P-Worm – 179, Trojan - 277. Таким образом, суммарно исследуемая выборка составила 1 369 отчётов. Нижняя граница длины паттерна установлена равной 3 событиям.

Для анализа были выбраны алгоритмы поиска закрытых последовательных паттернов CloSpan и ClaSP. В этих алгоритмах представлены разные методы поиска. CloSpan основан на представлении исходных данных в виде дерева. ClaSP использует вертикальное внутреннее представление исходных данных и обладает большей скоростью работы [4].

Количественные показатели закрытых паттернов 1-го уровня обнаруженных алгоритмом CloSpan для каждого класса вредоносных программ следующие. Для Backdoor при объёме выборки 595 и поддержке 70 % получено 49 паттернов, при поддержке 60 % – 389 паттернов. Для Virus при объёме выборки 94 и поддержке 70 % получено 2 паттерна, при поддержке 60 % – 9 паттернов, при поддержке 50 % – 12 паттернов. Для Worm при объёме выборки 224 и поддержке 70 % – 11 паттернов, при поддержке 60 % – 90 паттернов, при поддержке 50 % – 998 паттернов. Для P2P-Worm при объёме выборки 179 и поддержке 70 % – 253, при поддержке 60 % – 688. Для Trojan при объёме выборки 277 и поддержке 70 % получено 38 паттернов, при поддержке 60 % – 243 паттерна.

Количественные показатели закрытых паттернов, обнаруженных алгоритмом ClaSP, для каждого класса вредоносных программ следующие. Для Backdoor при объёме выборки 595 и поддержке 70 % получено 49 паттернов, при поддержке 60 % – 369 паттернов. Для Virus при объёме выборки 94 и поддержке 70 % получено 5 паттернов, при поддержке 60 % – 14 паттернов, при поддержке 50 % – 26 паттернов. Для Worm при объёме выборки 224 и поддержке 70 % – 11 паттернов, при поддержке 60 % – 90 паттернов, при поддержке 50 % – 998 паттернов. Для P2P-Worm при объёме выборки 179 и поддержке 70 % – 253, при поддержке 60 % – 688, при поддержке 50 % – 1268. Для Trojan при объёме выборки 277 и поддержке 70 % получено 71 паттерн, при поддержке 60 % – 318 паттернов.

В следующем примере представлен паттерн, полученный из выборки отчётов о поведении вирусов:

GetProcAddress()–InitializeSecurityDescriptor()– SetSecurityDescriptorDacl()–FreeSid()–GetProcAddress()

Он демонстрирует работу вируса с дескриптором безопасности некоторого процесса.

Другой пример найденных паттернов – для семейства Agent класса Backdoor состоял из следующих паттернов:

1. GetACP – GetProcAddress –LoadLibraryA.
Паттерн соответствует получению кодовой страницы компьютера.

2. GetProcAddress – InitializeAcl – AddAccessAllowedAce – InitializeSecurityDescriptor – RegCreateKeyExA – GetProcAddress – LoadLibraryA.

Паттерн демонстрирует, как вредоносная программа-бэкдор создает список ограничений (в данном случае – одно ограничение доступа с помощью дескриптора безопасности) и ставит его на ключ реестра.

3. GetProcAddress – AllocateAndInitializeSid – InitializeAcl – AddAccessAllowedAce – InitializeSecurityDescriptor – RegCreateKeyExA – GetProcAddress – LoadLibraryA.

Паттерн демонстрирует, как вредоносная программа-бэкдор добавляет ограничение в созданный список безопасности, создает дескриптор безопасности и все это подключает к ключу реестра.

4. GetProcAddress – AllocateAndInitializeSid – InitializeAcl – AddAccessAllowedAce – InitializeSecurityDescriptor – RegCreateKeyExA – FreeSid – GetProcAddress – LoadLibraryA .

Данный паттерн демонстрирует действия, аналогичные предыдущему паттерну, но кроме того здесь бэкдор освобождает дескриптор безопасности.

Проведен эксперимент по поиску паттернов 2-го уровня. В качестве выборки D_p^1 взята выборка максимальных без повторов паттернов, найденных в поведении вредоносов классов P2P-Worm и Trojan алгоритмом ClaSP при поддержке 50 %. Это самая объёмная результирующая выборка паттернов, полученных на этапе поиска: 1268 образцов для P2P-Worm и 318 для Trojan.

Результаты эксперимента для класса P2P-Worm на этапе получения выборки D_p^2, то есть выборки паттернов 2-го уровня для алгоритма ClaSP: 1 двухуровневый паттерн для величины поддержки 40 %, 2 двухуровневых паттерна для поддержки 30 %. Для алгоритма CloSpan: 1 двухуровневый паттерн для величины поддержки 50 %, 2 двухуровневых паттерна для поддержки 30 %.

Результаты эксперимента для класса Trojan на этапе получения выборки D_p^2 для алгоритма ClaSP: 1 двухуровневый паттерн для величины поддержки 50 %, 3 двухуровневых паттерна для поддержки 30 %. Для алгоритма CloSpan: 1 двухуровневый паттерн для величины поддержки 50 %, 4 двухуровневых паттерна для поддержки 30 %.

Из результатов видно, что количество паттернов 2-го уровня заметно ниже, чем для 1-го уровня. Таким образом, в паттернах 2-го уровня, как и предполагалось, сгруппированы наиболее значимые паттерны 1-го уровня.

Рассмотрим для примера один из паттернов 2-го уровня, полученный для вредоносных программ класса Trojan. Он состоит из двух паттернов 1-го уровня:

1. Паттерн p_1^1: RegOpenKeyExW() - LoadLibraryA() - RegOpenKeyExA() - LocalFree() - RegCreateKeyExA() - GetSystemMetrics() - GetModuleFileNameA().

2. Паттерн p_2^1: RegOpenKeyExW() - LoadLibraryA() - RegOpenKeyExA()- LocalFree() - RegCreateKeyExA() - GetModuleFileNameA() - GetVersion().

Приведенные паттерны демонстрируют, как троянские программы внедряются в систему с помощью редактирования реестра, пытаясь делать это последовательно двумя разными способами. Схожее поведение демонстрирует другой паттерн 2-го уровня, состоящий из трёх паттернов 1-го уровня:

1. Паттерн p_1^1 RegOpenKeyExW() - LoadLibraryA() - RegOpenKeyExA() - LocalFree() - RegCreateKeyExA() - LoadLibraryA() - RegCloseKey().

2. Паттерн p_2^1 RegOpenKeyExW() - LoadLibraryA() - RegOpenKeyExA() - LocalFree() - RegCreateKeyExA() - RegOpenKeyExW() - LoadLibraryA().

3. Паттерн p_3^1 RegOpenKeyExW() - LoadLibraryA() - RegOpenKeyExA() - LocalFree() - RegCreateKeyExA() - LoadLibraryA() - RegOpenKeyExA().

Таким образом, найденные паттерны показывают разные комбинации действий, применяемые последовательно, которыми троянские программы пытаются достигнуть одной и той же цели.

Выводы

В данной работе представлен метод получения многоуровневых последовательных паттернов, использующий методы секвенциального анализа и регулярные выражения. Выдвинуты требования к практической реализации предложенного метода в рамках программной системы.

Продемонстрированы результаты экспериментов по секвенциальному анализу данных о поведении вредоносных программ с использованием предложенных методов. В результате экспериментов были получены паттерны первого и второго уровня. На основе двух выборок паттернов первого уровня объёмом 1268 и 318 образцов получено паттернов второго уровня для двух классов при поддержке 50 % – от 1 до 2, при поддержке 30 % от 2 до 4 для каждого класса. Полученные паттерны можно использовать для изучения поведения вредоносных программ и для классификации новых вредоносных программ.

1. *Agrawal R., Srikant R.* Mining sequential patterns. – Data Engineering. – 1995. – С. 3–14.
2. *Gupta M., Han J.* Approaches for pattern discovery using sequential data mining. – Pattern Discovery Using Sequence Data Mining: Applications and Studies. – IGI Global. – 2012. – С. 137–154.
3. *Feida Zhu, Xifeng Yan, Jiawei Han and Philip S.Yu.* Efficient discovery of frequent approximate sequential patterns. – ICDM '07: Proceedings of the Seventh IEEE International Conference on Data Mining – Washington, DC, USA, IEEE Computer Society. – 2007. – С. 751–756.
4. *Mabroukeh N.R., Ezeife C.I.* A taxonomy of sequential pattern mining algorithms // ACM Computing Surveys (CSUR). – 2010. – Т. 43, N 1. – P. 3.
5. *Fournier-Viger P., Gomariz Gueniche T.A., Soltani A., Wu., C., Tseng V.S.* SPMF: a Java Open-Source Pattern Mining Library // Journal of Machine Learning Research (JMLR), 15. – 2014. – P. 3389–3393.
6. *Chen Zhongqiang, et al.* Malware characteristics and threats on the internet ecosystem // Journal of Systems and Software 85.7. – 2012. – P. 1650–1672.
7. *Sami A. et al.* Malware detection based on mining API calls // Proceedings of the 2010 ACM Symposium on Applied Computing. – ACM, 2010. – P. 1020–1025.

References

1. Srikant, R. and Agrawal, R. (1996). Mining sequential patterns. San Jose [etc.]: IBM Thomas J. Watson Research Division.
2. Gupta, M. and Han, J. (n.d.). Approaches for Pattern Discovery Using Sequential Data Mining. Pattern Discovery Using Sequence Data Mining, pp.137-154.
3. Zhu, F., Yan, X., Han, J. and Yu, P. (2007). Efficient Discovery of Frequent Approximate Sequential Patterns. Seventh IEEE International Conference on Data Mining (ICDM 2007), pp.751 - 756.
4. Mabroukeh, N. and Ezeife, C. (2010). A taxonomy of sequential pattern mining algorithms. CSUR, 43(1), pp.1-41.
5. Fournier-Viger, P et. al. (2014) SPMF: a Java open-source pattern mining library. J. Mach. Learn. Res. 15, 1, 3389-3393.
6. Chen, Z., Roussopoulos, M., Liang, Z., Zhang, Y., Chen, Z. and Delis, A. (2012). Malware characteristics and threats on the internet ecosystem. Journal of Systems and Software, 85(7), pp.1650-1672.
7. Sami, A., Yadegari, B., Peiravian, N., Hashemi, S. and Hamze, A. (2010). Malware detection based on mining API calls. Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10, pp.1020-1025.

Об авторе:

Молдавская Александра Владимировна,
аспирант Одесского национального политехнического университета.
Количество научных публикаций в украинских изданиях – 4.
Индекс Гирша – 0.
http://orcid.org/0000-0002-1079-5850

Место работы автора:

Одесский национальный политехнический университет
65044, Украина, Одесса, пр. Шевченко, 1.
Тел./факс: +38 (048) 705-8301, +38 (048) 705-8302.
E-mail: opu@opu.ua, amme4od@mail.ru