НОРМАЛИЗАЦИЯ ЖУРНАЛОВ СОБЫТИЙ С ИСПОЛЬЗОВАНИЕМ ДЕРЕВА ФИКСИРОВАННОЙ ГЛУБИНЫ

Москвичев А.Д.[1], Долгачев М.В.[2]

Цель статьи: *разработка программного средства для нормализации журналов событий, использующегося в качестве модуля систем управления информацией о безопасности и событиями безопасности.*

Метод: *нормализация журналов событий с использованием дерева фиксированной глубины, поскольку такой метод дает высокую скорость обработки входных данных при низкой вероятности ложных срабатываний, однако требует написания регулярных выражений.*

Полученный результат: *описан алгоритм нормализации журналов событий, использующий в своей работе дерево фиксированной глубины. Приведено сравнение с другими методами нормализации журналов событий по уровню точности. Разработано программное средство, реализующее данный алгоритм. Произведено тестирование полученного программного средства на реальных данных, вычислено время обработки одного события, сделан вывод о средне возможном количестве событий, обрабатываемом в секунду. Авторы статьи приводят схему интеграции полученного программного средства с системой управления информацией о безопасности и событиями безопасности. В заключении приведены оценки об эффективности самого алгоритма, а также программного средства, полученного на основе описанного алгоритма нормализации журналов событий, в качестве модуля системы управления информацией о безопасности и событиями безопасности.*

Ключевые слова: *информационная безопасность, журнал событий, анализ инцидентов, мониторинг событий, статистический алгоритм, инцидент информационной безопасности, синтаксический анализ.*

DOI: 10.21681/2311-3456-2020-2-76-82

1. Введение

Журнал событий — стандартный способ для программных средств, операционной системы и сетевых устройств записи и централизованного хранения информации о важных программных и аппаратных событиях. Анализ журналов событий является необходимым инструментом для выявления инцидентов в сфере информационной безопасности [1].

С постоянно растущим масштабом и сложностью современных систем объем журналов быстро растет. Поэтому традиционный способ анализа журналов, который в значительной степени опирается на ручную проверку, стал трудоемкой и подверженной ошибкам задачей. Чтобы решить эту проблему, в последнее время было предпринято много попыток автоматизировать анализ журналов с использованием методов интеллектуального анализа данных. Однако необработанные сообщения журнала обычно не структурированы, поскольку разработчикам разрешается записывать сообщения журнала в свободной форме для удобства и гибкости. Чтобы применить автоматическое извлечение неструктурированных журналов, первым делом необходимо выполнить синтаксический анализ, посредством чего неструктурированные необработанные сообщения могут быть преобразованы в последовательность структурированных событий [2].

Для достижения этой цели был предложен ряд управляемых данными подходов для автоматического анализа журналов. Однако готовых реализаций инструментальных средств синтаксического анализа практически не существует. Даже с коммерческими решениями по управлению журналами, являющимися частью систем управления информацией о безопасности и событиями безопасности (далее SIEM — «Security information and event management» [3]), пользователи должны предоставлять сложные конфигурации с настраиваемыми правилами для анализа своих журналов [4]. В данном случае инженеры и аналитики должны реализовывать свои собственные «парсеры» журналов при выполнении задач анализа, что потребует много времени.

2. Алгоритм Drain

Drain — метод анализа журналов событий в режиме реального времени с использованием дерева с фиксированной глубиной [5]. При получении нового

1 Долгачев Михаил Владимирович, кандидат технических наук, доцент, ФГБОУ ВО «Тихоокеанский государственный университет», г. Хабаровск, Россия. E-mail: 007428@pnu.edu.ru

2 Москвичев Антон Дмитриевич, аспирант, ФГБОУ ВО «Тихоокеанский государственный университет», г. Хабаровск, Россия. E-mail: anton.moskvichev.1996@yandex.ru

необработанного сообщения журнала алгоритм будет предварительно обрабатывать его с помощью простых регулярных выражений, основанных на знании предметной области. Затем происходит поиск группы журналов (то есть конечный узел дерева), следуя специально разработанным правилам, закодированным во внутренних узлах дерева. Если найдена подходящая группа журналов, сообщение журнала будет сопоставлено с событием, сохраненным в этой группе. В противном случае будет создана новая группа на основе полученного сообщения.

Когда поступает необработанное сообщение, анализатор журналов должен найти для него наиболее подходящую группу или создать новую. Простым решением этой задачи является сравнение необработанного сообщения с событием журнала, хранящимся в каждой группе, один за другим. Однако это решение очень медленное, потому что количество групп журналов быстро увеличивается. Чтобы ускорить этот процесс, разрабатывается дерево разбора с фиксированной глубиной, чтобы вести поиск в группе журналов, которая ограничивает количество групп журналов, с которыми необходимо сравнить необработанное сообщение.

Корневой узел и внутренние узлы кодируют специально разработанные правила для управления процессом поиска. Они не содержат групп журналов. Каждый путь в дереве синтаксического анализа заканчивается конечным узлом, в котором хранится список групп журналов. Каждая группа журналов состоит из двух частей: шаблон события и идентификатор журнала. Шаблон события — это шаблон, состоящий из постоянной части сообщения журнала, который лучше всего описывает сообщения журнала в этой группе. Идентификаторы журналов представляют собой идентификаторы сообщений журнала в этой группе. Глубина всех конечных узлов одинакова и фиксируется заданной величиной.

Алгоритм включает в себя следующие шаги:

Шаг 1: предварительная обработка. Перед использованием дерева синтаксического анализа сообщение журнала обрабатывается при его поступлении с помощью регулярных выражений. Инженер заранее должен предоставить простые регулярные выражения, которые представляют собой используемые переменные, такие как IP-адрес. Затем алгоритм удалит поля, сопоставленные с сообщением необработанного журнала этими регулярными выражениями.

Регулярные выражения, используемые на этом шаге, должны быть просты, чтобы увеличить скорость обработки сообщений. Обычно требуется не более трех таких регулярных выражений.

Шаг 2: поиск по длине сообщения журнала. Узлы первого слоя в дереве синтаксического анализа хранят длины сообщений, и предназначены для разделения журналов на группы с различной длиной. Под длиной сообщения журнала понимается количество полей в сообщении. На этом этапе алгоритм выбирает путь к узлу первого слоя на основе длины предварительно обработанного сообщения. Использование такого подхода основано на предположении, что сообщения журнала с одним и тем же событием, вероятно, будут иметь одинаковую длину. Хотя возможно, что сообщения журнала с одним и тем же событием имеют различную длину, они могут быть обработаны посредством постобработки.

Шаг 3: поиск по полям. На этом этапе алгоритм проходит от узла первого слоя, поиск которого выполняется на этапе 2, к конечному узлу. Этот шаг основан на предположении, что поля в начальных позициях сообщения журнала чаще всего являются постоянными. В частности, алгоритм выбирает следующий внутренний узел с помощью полей в начальных позициях сообщения журнала.

Шаг 4: поиск шаблона. Перед этим шагом алгоритм перешел к конечному узлу, который содержит список шаблонов. Сообщения журнала в этих группах журналов соответствуют правилам, закодированным во внутренних узлах по пути.

На этом шаге алгоритм выбирает наиболее подходящий шаблон из списка, вычисляя коэффициент подобия между сообщением журнала и шаблонами:

$$simSeq = \frac{\sum_{i=1}^{n} equ(seq_1(i), seq_2(i))}{n}, \quad (1)$$

где: seq_1 и seq_2 представляют соответственно сообщение журнала и событие журнала; i — номер символа в последовательности; N — длина сообщения; функция equ определяется следующим образом:

$$equ(t_1, t_2) = \begin{cases} 1 \text{ если } t_1 = t_2 \\ 0 \text{ иначе} \end{cases}. \quad (2)$$

Здесь t_1 и t_2 — два токена. Найдя группу журналов с наибольшим коэффициентом, происходит сравнение её с предопределённым порогом подобия st. Если $simSeq \geq st$, алгоритм возвращает шаблон как наиболее подходящий. В противном случае алгоритм возвращает флаг, указывающий на отсутствие подходящего шаблона.

Шаг 5: обновление синтаксического дерева. Если на шаге 4 будет возвращен подходящий шаблон, алгоритм добавит идентификатор журнала текущего сообщения к идентификаторам журнала в группе, соответствующей шаблону.

Если алгоритм не может найти шаблон, он создает новую группу на основе текущего сообщения журнала, где идентификаторы журналов содержат только идентификатор сообщения, а событие является именно сообщением журнала. Затем алгоритм обновит дерево синтаксического анализа с помощью новой группы журналов. Алгоритм проходит от корневого узла к конечному, который должен содержать новую группу журналов, и добавляет отсутствующие внутренние узлы и конечный узел соответственно вдоль пути.

3. Точность алгоритма

Точность показывает, насколько правильно анализатор журнала сопоставляет необработанные сообщения с нормализованными, и определяется как отношение количества верно обработанных журналов

событий к количеству всех событий. Точность важна, поскольку ошибки синтаксического анализа могут ухудшить производительность последующей задачи анализа журналов. Методы синтаксического анализа журналов в режиме реального времени имеют более высокую точность по сравнению с другими методами, поскольку они используют все необработанные сообщения журнала в начале синтаксического анализа, в то время как иные корректируют свою модель постепенно.

Для сравнения можно оценить точность четырех методов синтаксического анализа журналов для различных наборов данных:

1. LKE: Это метод синтаксического анализа журналов, разработанный корпорацией Microsoft. В нем используется иерархическая кластеризация и эвристические правила.
2. IPLoM: IPLoM выполняет трехэтапное иерархическое разбиение перед созданием шаблона.
3. SHISO [6]: В этом синтаксическом анализаторе дерево с предварительно определенным числом потомков в каждом узле используется для управления поиском групп журналов.
4. Spell: Этот метод использует самую длинную общую последовательность для поиска группы журналов. Он ускоряет процесс поиска за счет подпоследовательности сопоставления и дерева префиксов.

Наборы данных, представляющие собой ненормализованные журналы событий, приведены в таблице 2. Результаты расчета точности каждого из алгоритмов для каждого набора данных представлены в таблице 3. В таблице 4 демонстрируется время нормализации событий из наборов данных для каждого из рассматриваемых алгоритмов в секундах.

Таблица 2

Наборы данных для анализа

Система	Описание	Количество сообщений в журнале	Количество событий
BGL	BlueGene / L Суперкомпьютер	4 474 963	376
HPC	Высокопроизводительный кластер	433 490	105
HDFC	Файловая система Hadoop	11 175 629	29
Zookeeper	Распределённый системный координатор	74 380	80
Proxifier	Прокси-клиент	10 108	8

Таблица 3

Точность работы алгоритмов

	BGL	HPC	HDFS	Zookeeper	Proxifier
LKE	0,67	0,17	0,57	0,78	0,85
IPLoM	0,99	0,65	0,99	0,99	0,85
SHISO	0,87	0,53	0,93	0,68	0,85
Spell	0,98	0,82	0,87	0,99	0,87
Drain	0,99	0,84	0,99	0,99	0,86

Таблица 4

Время выполнения работы алгоритмов

	BGL	HPC	HDFS	Zookeeper	Proxifier
LKE	N/A	N/A	N/A	N/A	8888,49
IPLoM	140,57	12,74	333,03	2,17	0,38
SHISO	10964,55	582,14	6649,23	87,61	8,41
Spell	447,14	47,28	676,45	5,27	0,87
Drain	115,96	8,76	325,7	1,81	0,27

LKE не может обрабатывать наборы данных, за исключением Proxifier, поскольку его сложность по времени $O(n^2)$, делает его слишком медленным.

Можно сделать вывод, что предлагаемый метод анализа в режиме реального времени, обеспечивает наилучшую точность для четырех наборов данных. LKE не так хорош для некоторых наборов данных, поскольку он использует неоптимальный метод кластеризации. IPLoM имеет высокую точность для большинства наборов данных благодаря своим специально разработанным правилам. SHISO использует подобие символов в сообщениях журнала для поиска соответствующих событий журнала. Этот метод слишком грубый, что вызывает неточности. Spell является точным, но его метод, основанный только на самой длинной общей подпоследовательности, приводит к неточностям.

4. Пример работы алгоритма

Программное средство, реализующее работу алгоритма, написано на компилируемом многопоточном языке программирования Go с использованием стандартных библиотек [7-9]. Для тестирования работоспособности алгоритма взято более 24000 ненормализованных событий, полученных от системы обнаружения вторжений Suricata в формате syslog [10, 11]. Заранее заданные параметры представлены в таблице 1. Результат работы программного средства представлен на рисунке 1.

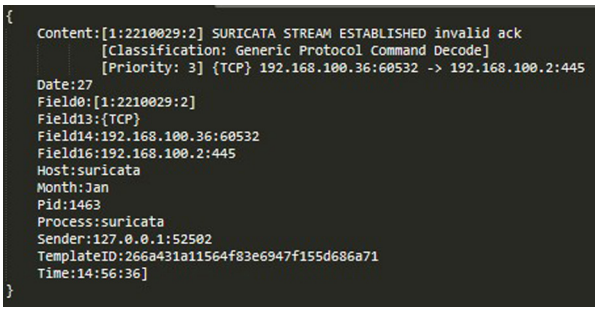

Рисунок 1. Результат работы программного средства

Передача журналов событий осуществлялась по протоколу UDP. На обработку одного события необходимо в среднем $18 \cdot 10^{(-6)}$ секунд, что эквивалентно $55 \cdot 10^3$ событиям в секунду.

5. Интеграция с SIEM-системой

Алгоритм одновременно может работать только с одним набором параметров, в связи с этим для интеграции с SIEM-системой в качестве модуля нормализации предлагается использовать несколько деревьев в разных потоках. Каждому потоку соответствует свой набор параметров.

Для отправки события в нужный поток необходимо ввести идентификатор, который будет однозначно определять, в какой поток необходимо передать полученное событие. Например, для событий, получаемых от системы обнаружения вторжений Suricata, идентификатором события выступает имя процесса, записываемого в формате syslog — «suricata».

Процесс обработки событий модулем нормализации представлен на рисунке 2. Первым этапом происходит инициализация: модуль управления получает из базы данных заранее прописанные параметры, необходимые в работе алгоритма, и инициализирует потоки в соответствии с заданными параметрами. Рекомендуется использовать документоориентированну систему управления базами данных [12], например, MongoDB [13] или Elasticsearch [14, 15]. При получении события от источника модуль управления передает его в поток в соответствии с идентификатором. Поток возвращает в модуль управления нормализованное событие в формате JSON, который, в свою очередь, передает событие брокеру сообщений для дальнейшего анализа SIEM-системой. Чаще всего используются такие брокеры сообщений, как RabbitMQ [16, 17], Apache Kafka [18, 19], Redis [20] и другие.

Таблица 1

Параметры алгоритма

Имя	Значение	Пояснение
Формат журнала	(?P<Month>[A-Z][a-z]{2})\s+(?P<Date>[0-9]{1,2})\s+(?P<Time>[0-9]{2}:[0-9]{2}:[0-9]{2})\s+(?P<Host>\w+)\s+(?P<Process>[0-9A-Za-z_-]+)(\[(?P<Pid>[0-9]+)\])?:\s+(?P<Content>.*)	Общий для syslog формат событий
Регулярные выражения	`\[[0-9]+:[0-9]+:[0-9]+\]`, `\{TCP\}|\{UDP\}`	Записываются для полей из общего поля Content согласно формату журнала
Коэффициент подобия	0.39	Необходим в шаге 4
Глубина дерева	4	

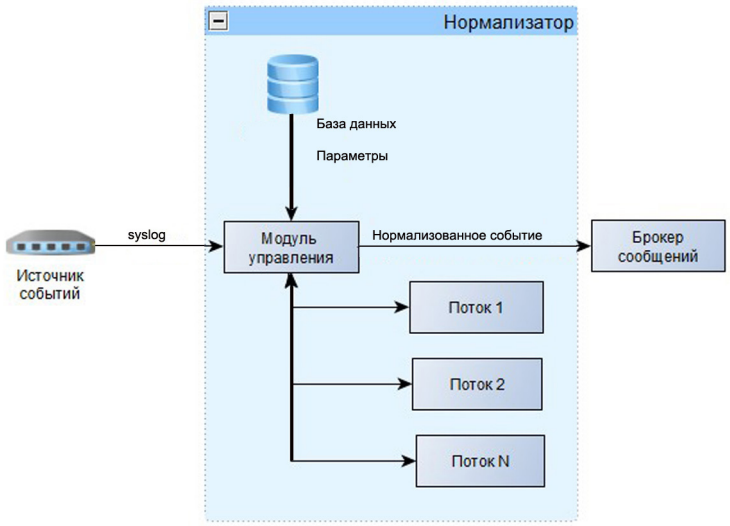

Рисунок 2. Процесс обработки событий модулем нормализации

6. Выводы

Синтаксический анализ журналов имеет большое значение для методов мониторинга и управления информационной безопасностью на основе анализа журналов. В этой статье предлагается метод анализа журнала в режиме реального времени Drain, который использует дерево синтаксического анализа фиксированной глубины для ускорения процесса поиска группы журналов. Для оценки эффективности алгоритма приводятся эксперименты по пяти наборам данных. Экспериментальные результаты показывают, что он значительно превосходит существующие анализаторы журналов с точки зрения точности и эффективности.

Алгоритм Drain можно использовать как модуль для нормализации журналов событий в продуктах класса SIEM. Однако, необходимо учесть, что алгоритм не может корректно идентифицировать найденные поля с точки зрения симптоматики. Поэтому для корректной работы необходима постобработка получаемых от него событий.

Рецензент: *Бегаев Алексей Николаевич, кандидат технических наук, Генеральный директор АО «Эшелон-СЗ», Санкт-Петербург, Россия. E mail: a.begaev@nwechelon.ru*

Литература

1. Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, B. Hu, Everything as a service (xaas) on the cloud: origins, current and future trends // Proc. of the 8th International Conference on Cloud Computing, 2015, pp. 621–628.
2. Бирюков А. А. Информационная безопасность: защита и нападение / А. А. Бирюков. 2-е изд., перераб. и доп. М. : ДМК Пресс, 2017. 434 с.
3. Абденов А.Ж. Анализ, описание и оценка функциональных узлов SIEM-системы : учебное пособие / Абденов А.Ж., Трушин В.А., Сулайман К. Электрон. текстовые данные. Новосибирск: Новосибирский государственный технический университет, 2018. 122 с.
4. D. Q. Zou, H. Qin, H. Jin, Uilog: Improving log-based fault diagnosis by log analysis // Journal of Computer Science and Technology, vol. 31, no. 5, pp. 1038–1052, 2016.
5. P. He, J. Zhu, Z. Zheng, M. R. Lyu, Drain: An online log parsing approach with fixed depth tree // ICWS, 2017, pp. 33–40
6. M. Du and F. Li, Spell: Streaming parsing of system event logs // Proc. of the 16th International Conference on Data Mining, 2016. DOI: 10.1109/ICDM.2016.0103
7. Батчер М. Go на практике / Мэтт Батчер, Мэтт Фарина ; пер. с англ. Р. Н. Рагимова; науч. ред. А. Н. Киселев. М.: ДМК Пресс, 2017. 374 с.
8. Donovan A., Kernighan B. The Go Programming Language / Alan A. A. Donovan, Brian W. Kernighan – Boston, USA : Addison–Wesley, 2015. 380 p.
9. Cox-Buday K. Concurrency in Go. Tools and Techniques for Developers / Katherine Cox-Buday – Sebastopol, USA : O'Reilly Media, 2017. 229 p.
10. Петренко С. А. Политики безопасности компании при работе в Интернет / С. А. Петренко, В. А. Курбатов. 3-е изд. (эл.). Электрон. текстовые дан. (1 файл pdf : 397 с.). М. : ДМК Пресс, 2018.

11. Петренко С. А. Управление информационными рисками. Экономически оправданная безопасность / С. А. Петренко, С. В. Симонов. 2-е изд. (эл.). Электрон. текстовые дан. (1 файл pdf : 396 с.). М. : ДМК Пресс, 2018, 384 с. ISBN 5-98453-001-5
12. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. М. : СОЛОН-Пресс, 2015. 320 с.
13. Chodorow K., Bradshaw S., Brazil E. MongoDB: The Definitive Guide, 3rd Edition / Kristina Chodorow, Shannon Bradshaw, Eoin Brazil – Sebastopol, USA : O'Reilly Media, 2019. 514 p.
14. Gormley C., Tong Z. Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine / C. Gormley, Z. Tong — Sebastopol, USA : O'Reilly Media, 2015. 724 p.
15. Chhajed S. Learning ELK Stack / Saurabh Chhajed – Birmingham, UK : Packt Publishing, 2015. 206 p.
16. Toshev M. Learning RabbitMQ / Martin Toshev – Birmingham, UK : Packt Publishing, 2015. 262 p.
17. Ayanoglu E., Aytas Y., Nahum D. Mastering RabbitMQ / Emrah Ayanoglu, Yusuf Aytas, Dotan Nahum – Birmingham, UK : Packt Publishing, 2016. 286 p.
18. Minni S. Apache Kafka Cookbook / Saurabh Minni – Birmingham, UK : Packt Publishing, 2015. 128 p.
19. Garg N. Learning Apache Kafka, 2nd Edition / Nishant Garg – Birmingham, UK : Packt Publishing, 2015. 112 p.
20. Huang P., Wang Z. Redis 4.x Cookbook / Pengcheng Huang, Zuofei Wang – Birmingham, UK : Packt Publishing, 2018. 382 p.

NORMALIZING EVENT LOGS USING A FIXED DEPTH TREE

Moskvichev A. D.[3], Dolgachev M. V.[4]

Purpose of the article: *development of a software tool for normalizing event logs, which is used as a module for managing security information and security events.*

Method: *normalization of event logs using a tree of fixed depth, since this method gives a high speed of processing input data with a low probability of false positives, however, it requires writing regular expressions.*

The result: *an algorithm for normalizing event logs is described that uses a fixed depth tree in its work. A comparison is made with other methods of normalizing event logs in terms of accuracy. A software tool has been developed that implements this algorithm. The obtained software was tested on real data, the processing time of one event was calculated, and a conclusion was drawn about the average possible number of events processed per second. The authors of the article give a scheme for integrating the resulting software with a system for managing security information and security events. In the conclusion, estimates are given about the effectiveness of the algorithm itself, as well as the software obtained on the basis of the described algorithm for normalizing event logs, as a module of a system for managing security information and security events.*

Keywords: *information security, event log, incident analysis, event monitoring, statistical algorithm, information security incident, parsing.*

References

1. Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, B. Hu, Everything as a service (xaas) on the cloud: origins, current and future trends // Proc. of the 8th International Conference on Cloud Computing, 2015, pp. 621–628.
2. Biriukov A. A. Informatcionnaia bezopasnost`: zashchita i napadenie / A. A. Biriukov. 2-e izd., pererab. i dop. M. : DMK Press, 2017. 434 s.
3. Abdenov A.ZH. Analiz, opisanie i ocenka funktcional`ny`kh uzlov SIEM-sistemy` : uchebnoe posobie / Abdenov A.ZH., Trushin V.A., Sulai`man K. E`lektron. tekstovy`e danny`e. Novosibirsk: Novosibirskii` gosudarstvenny`i` tekhnicheskii` universitet, 2018. 122 c.
4. D. Q. Zou, H. Qin, H. Jin, Uilog: Improving log-based fault diagnosis by log analysis // Journal of Computer Science and Technology, vol. 31, no. 5, pp. 1038–1052, 2016.
5. P. He, J. Zhu, Z. Zheng, M. R. Lyu, Drain: An online log parsing approach with fixed depth tree // ICWS, 2017, pp. 33–40
6. M. Du and F. Li, Spell: Streaming parsing of system event logs // Proc. of the 16th International Conference on Data Mining, 2016. DOI: 10.1109/ICDM.2016.0103
7. Batcher M. Go na praktike / Me`tt Batcher, Me`tt Farina ; per. s angl. R. N. Ragimova; nauch. red. A. N. Kiselev. M.: DMK Press, 2017. 374 s.
8. Donovan A., Kernighan B. The Go Programming Language / Alan A. A. Donovan, Brian W. Kernighan – Boston, USA : Addison–Wesley, 2015. 380 p.

3 Anton Moskvichev, postgraduate, Pacific National University, Khabarovsk, Russia. E-mail: anton.moskvichev.1996@yandex.ru
4 Mihail Dolgachev, Ph. D. (in Tech.), Pacific National University, Khabarovsk, Russia. E-mail: 007428@pnu.edu.ru

9. Cox-Buday K. Concurrency in Go. Tools and Techniques for Developers / Katherine Cox-Buday – Sebastopol, USA : O'Reilly Media, 2017. 229 p.
10. Petrenko S. A. Politiki bezopasnosti kompanii pri rabote v Internet / S. A. Petrenko, V. A. Kurbatov. 3-e izd. (e`l.). E`lektron. tekstovy`e dan. (1 fai`l pdf : 397 s.). M. : DMK Press, 2018.
11. Petrenko S. A. Upravlenie informatcionny`mi riskami. E`konomicheski opravdannaia bezopasnost` / S. A. Petrenko, S. V. Simonov. 2-e izd. (e`l.). E`lektron. tekstovy`e dan. (1 fai`l pdf : 396 s.). M. : DMK Press, 2018, 384 c. ISBN 5-98453-001-5
12. Tarasov, S. V. SUBD dlia programmista. Bazy` danny`kh iznutri / S. V. Tarasov. M. : SOLON-Press, 2015. 320 s.
13. Chodorow K., Bradshaw S., Brazil E. MongoDB: The Definitive Guide, 3rd Edition / Kristina Chodorow, Shannon Bradshaw, Eoin Brazil – Sebastopol, USA : O'Reilly Media, 2019. 514 p.
14. Gormley C., Tong Z. Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine / C. Gormley, Z. Tong — Sebastopol, USA : O'Reilly Media, 2015. 724 p.
15. Chhajed S. Learning ELK Stack / Saurabh Chhajed – Birmingham, UK : Packt Publishing, 2015. 206 p.
16. Toshev M. Learning RabbitMQ / Martin Toshev – Birmingham, UK : Packt Publishing, 2015. 262 p.
17. Ayanoglu E., Aytas Y., Nahum D. Mastering RabbitMQ / Emrah Ayanoglu, Yusuf Aytas, Dotan Nahum – Birmingham, UK : Packt Publishing, 2016. 286 p.
18. Minni S. Apache Kafka Cookbook / Saurabh Minni – Birmingham, UK : Packt Publishing, 2015. 128 p.
19. Garg N. Learning Apache Kafka, 2nd Edition / Nishant Garg – Birmingham, UK : Packt Publishing, 2015. 112 p.
20. Huang P., Wang Z. Redis 4.x Cookbook / Pengcheng Huang, Zuofei Wang – Birmingham, UK : Packt Publishing, 2018. 382 p.