УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
май–июнь 2016 Том 16 № 3 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
May–June 2016 Vol. 16 No 3 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.6

МЕТОД ПРОВЕДЕНИЯ ПОСТИНЦИДЕНТНОГО ВНУТРЕННЕГО АУДИТА СРЕДСТВ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ НА ОСНОВЕ ГРАФОВ

И.С. Пантюхин[a], И.А. Зикратов[a], А.Б. Левина[a]

[a] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация
Адрес для переписки: zevall@cit.ifmo.ru

Информация о статье
Поступила в редакцию 29.05.15, принята к печати 16.04.16
doi: 10.17586/2226-1494-2016-16-3-506-512
Язык статьи – русский

Ссылка для цитирования: Пантюхин И.С., Зикратов И.А., Левина А.Б. Метод проведения постинцидентного внутреннего аудита средств вычислительной техники на основе графов // Научно-технический вестник информационных технологий, механики и оптики. 2016. Т. 16. № 3. С. 506–512. doi: 10.17586/2226-1494-2016-16-3-506-512

Аннотация

Предложен метод проведения постинцидентного внутреннего аудита средств вычислительной техники на основе графов. Сущность предлагаемого решения заключается в установлении взаимосвязей между дампами (образами) жесткого диска, оперативной памяти, сети. Метод предназначен для описания свойств инцидента информационной безопасности при проведении внутреннего постинцидентного аудита средств вычислительной техники. На первом шаге происходит процесс получения и формирование дампов жесткого диска, оперативной памяти и сети. Далее происходит разбор этих дампов на набор составляющих. Набор составляющих включает в себя большой набор атрибутов, которые составляют основу для формирования графа. Разобранные данные записываются в нереляционную систему управления базами данных (NoSQL), адаптированную для хранения, быстрого доступа и обработки графов. На заключительном шаге происходит установление взаимосвязей между дампами. Представленный метод позволяет человеку-эксперту в области информационной безопасности или компьютерной криминалистики проводить более точный, информативный внутренний аудит средств вычислительной техники. Предложенный метод позволяет снизить временные затраты на проведение внутреннего аудита средств вычислительной техники, повысить точность и информативность такого аудита. Метод имеет потенциал к развитию и может применяться в задачах идентификации пользователей и компьютерной криминалистике.

Ключевые слова

метод, внутренний аудит, информационная безопасность, графы, компьютерная криминалистика

GRAPH-BASED POST INCIDENT INTERNAL AUDIT METHOD OF COMPUTER EQUIPMENT

I.S. Pantiukhin[a], I.A. Zikratov[a], A.B. Levina[a]

[a] ITMO University, Saint Petersburg, 197101, Russian Federation
Corresponding author: zevall@cit.ifmo.ru

Article info
Received 29.05.15, accepted 16.04.16
doi: 10.17586/2226-1494-2016-16-3-506-512
Article in Russian

For citation: Pantiukhin I.S., Zikratov I.A., Levina A.B. Graph-based post incident internal audit method of computer equipment. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2016, vol. 16, no. 3, pp. 506–512. doi: 10.17586/2226-1494-2016-16-3-506-512

Abstract

Graph-based post incident internal audit method of computer equipment is proposed. The essence of the proposed solution consists in the establishing of relationships among hard disk damps (image), RAM and network. This method is intended for description of information security incident properties during the internal post incident audit of computer equipment. Hard disk damps receiving and formation process takes place at the first step. It is followed by separation of these damps into the set of components. The set of components includes a large set of attributes that forms the basis for the formation of the graph. Separated data is recorded into the non-relational database management system (NoSQL) that is adapted for graph storage, fast access and processing. Damps linking application method is applied at the final step. The presented method gives the possibility to human expert in information security or computer forensics for more precise, informative internal audit of computer equipment. The proposed method allows reducing the time spent on internal audit of computer equipment, increasing accuracy and informativeness of such audit. The method has a development potential and can be applied along with the other components in the tasks of users' identification and computer forensics.

Keywords

method, internal audit, information security, graphs, computer forensics

Введение

С ростом объема данных и средств вычислительной техники количество инцидентов информационной безопасности растет [1]. Большинство из этих инцидентов, возникающих в процессе эксплуатации, остаются неисследованными из-за отсутствия современных методов и подходов комплексного, внутреннего аудита, отвечающих современным требованиям развития техники. В связи с этим встают задачи разработки методов, способов, подходов, которые бы позволяли снизить временные затраты внутреннего аудита и вероятность ошибки анализа данных, а также повысить информативность инцидента информационной безопасности и точность идентификации средств вычислительной техники.

Под постинцидентным внутренним аудитом в настоящей работе подразумевается восстановление событий инцидентов информационной безопасности, произошедших на средстве вычислительной техники. Постинцидентный внутренний аудит преследует цель получения данных со средства вычислительной техники и поиска в них информации об инцидентах, которые могут находиться в дампах жесткого диска, оперативной памяти и сети.

На данный момент задачи внутреннего аудита данных со средства вычислительной техники (жесткого диска, оперативной памяти, сети) решают отдельно друг от друга. Основные способы получения данных и их аудита описаны в работах [2–4]. С применением комплексного аудита этих данных, построением взаимосвязей между ними [5], с применением поисковых алгоритмов [6] появляется возможность расследовать инцидент информационной безопасности, а также снизить временные затраты, повысить точность внутреннего аудита, снизить вероятность утраты цифровых улик.

В работе предложен метод постинцидентного внутреннего аудита средств вычислительной техники с использованием теории графов. Проведен эксперимент, подтверждающий работоспособность предложенного метода, по результатам которого сделаны заключения о временных затратах на проведение внутреннего аудита, определены результаты повышения информативности результатов расследования инцидента информационной безопасности.

Описание метода

Метод проведения постинцидентного внутреннего аудита средств вычислительной техники состоит из нескольких последовательных шагов. На первых шагах осуществляется процесс получения и формирования данных, которые состоят из дампов жесткого диска (HDD), оперативной памяти (RAM) [7, 8] и сети (NET) [9, 10]. Далее полученные данные разбираются на составляющие. Этот набор составляющих имеет большой набор атрибутов и их значений. После этапа разбора атрибуты и их значения записываются в нереляционную систему управления базами данных (NoSQL), адаптированную (специально сформированную) для хранения слабоструктурированных данных. Выбор нереляционной базы данных обусловлен необходимостью быстрой обработки слабоструктурированных данных и быстрого доступа к ним [11]. Процесс проведения постинцидентного внутреннего аудита изображен на рис. 1.

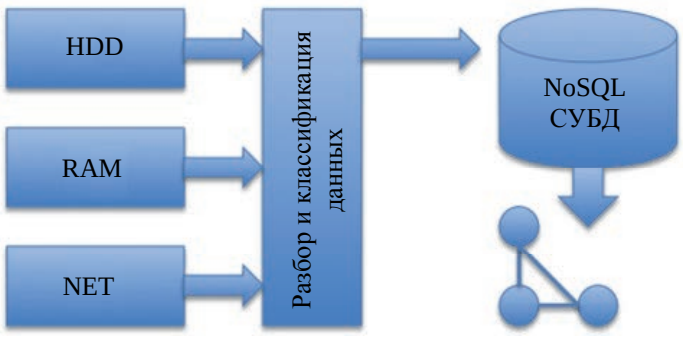

Рис. 1. Процесс проведения постинцидентного внутреннего аудита средства вычислительной техники

В результате разбора и классификации данных со средства вычислительной техники получается большое количество атрибутов с их значениями, например: хэш файла (Hash), имя файла (Name), полный путь к файлу (Directory), тип файла (Type), права файла (Permission), дата создания файла (Date), название процесса (NamePid), pid процесса (Pid), тип протокола (Protocol) и многие другие [12–14]. Эти атрибуты и их значения сохраняются в NoSQL СУБД и имеет структуру, показанную в табл. 1.

Для проведения постинцидентного внутреннего аудита необходимо описать взаимосвязи между атрибутами и их значениями, которые хранятся в NoSQL СУБД. Описать взаимосвязи между слабоструктурированными данными в NoSQL можно с использованием теории графов [15–18]. Она позволяет исследовать свойства конечных множеств с заданными отношениями между их элементами. Графом $G=(V, E)$ называется совокупность двух множеств – непустого множества V (множества вершин) и множества E двухэлементных подмножеств множества V (E – множество ребер). В нашем случае такими

множествами являются дампы жесткого диска (G_{hdd}), оперативной памяти (G_{ram}), сети (G_{net}), изображенные на рис. 1 как HDD, RAM и NET соответственно. Для построения графа, описывающего взаимосвязи между дампами, необходимо разработать методику, при помощи которой можно описывать атрибуты и их значения с дампов памяти в NoSQL СУБД. После построения и визуализации графа методика должна способствовать получению информации об инциденте при заданных перестроениях графа.

Hash	Name	Directory	Type	Permission	..
69b05b88007cca3e1870796f7b9e40ba	00001.vcf	/Users/zevall/Яндекс.Диск	vCard visiting card	-rw-r--r--	..
dbfa9f5eb5d502a647b54a7c910d46d5	datasci.png	/Users/zevall/Яндекс.Диск	PNG image data, 848 x 1200, 8-bit/color RGB, non-interlaced	-rw-r-----@	..
..	..	..	..	..	..

Таблица 1. Пример таблицы атрибутов данных и их значений

Методика построения взаимосвязей между данными (дампами) постинцидентного средства вычислительной техники

Рассмотрим три графа $G_{hdd} = (Vhdd, Ehdd)$, $G_{ram} = (Vram, Eram)$, $G_{net} = (Vnet, Enet$, где $Vhdd$, $Vram$, $Vnet$ – множества вершин соответствующих графов, а $Ehdd$, $Eram$, $Enet$ – множества ребер в каждом из этих графов. Каждый из этих трех графов является подграфом к $G = (V, E)$, $V = Vhdd \cup Vram \cup net$, $E = Ehdd \cup Eram \cup Enet \cup Ehr \cup Ern \cup Enh \cup Ehh \cup Err \cup Enn$, где Ehr – множество ребер между G_{hdd}и G_{ram}, Ern – между G_{ram}и G_{net}, Enh – между G_{net}и G_{hdd}, а Ehh, Err, Enn – множества ребер между вершинами, значения которых совпадают в подграфах G_{hdd}, G_{ram} и G_{net} соответственно.

После получения и формирования дампов жесткого диска, оперативной памяти и сети, а также разбора этих дампов на набор составляющих, имеющих огромный набор атрибутов, которые составляют основу для формирования графа, происходит построение изначального графа. Такой изначальный граф имеет множество вершин *V* и множество ребер *Ehdd*, *Eram*, *Enet*, все остальные ребра строятся по принципу сходства их вершин друг с другом. В рассматриваемых графах $|Vhdd| = n$; $|Vram| = m$; $|Vnet| = k$.

Пусть $Whdd$ – множество, в котором хранятся значения вершин G_{hdd}, аналогично и для $WRam$ и для $WNet$, тогда будем строить ребра следующим образом.

1. Преобразуем значения всех вершин $WHdd, WRam, WNet$ в значения одного типа, например, в числа:
   1.1 дальнейшая работа будет происходить не над множествами *WHdd*, *WRam*, *Wnet*, а над их копиями – множествами *Wh*, *Wr*, *Wn* соответственно;
   1.2 произведем сортировку элементов множеств по возрастанию либо по убыванию числовых значений *Sort*(*Wh*), *Sort*(*Wr*), *Sort*(*Wn*).
2. Поиск совпадающих значений проводим по следующему правилу.
   2.1. Сначала найдем ребра в каждом из подграфов по отдельности:
      2.1.1 для графа G_{hdd}будем искать совпадающие значения $\forall_{i,j} = 1..n : i \neq j, Wh_i = Wh_j$, после чего находим эти значения в исходном множестве значений $\exists lt, k = 1..\mathrm{n} : \mathrm{W}h_i = WHdd_t, Wh_i = WHdd_k$, $\{Vhdd_t, Vhdd_k\} \in Ehh$;
      2.1.2 аналогично для графа $G_{ram} \forall_{i,j} = 1..m : i \neq j, Wr_i = Wr_j, \exists_{t,k} = 1..m : Wr_i = WRam_t, Wr_j = WRam_k, \{Vram_t, Vram_k\} \in Err$;
      2.1.3 для графа $G_{net} \forall_{i,j} = 1..k : i \neq j$, $Wn_i = Wn_j, \exists_{t,z} = 1..k : Wn_i = WNet_t$, $Wn_i = WNet_z, \{Vnet_t, Vnet_z\} \in Enn$.
   2.2. Затем найдем ребра между каждой парой графов $G_{hdd}, G_{ram}, G_{net}$:
      2.2.1 для пары $G_{hdd}, G_{ram} - \forall_i = 1..n,\ j = 1..m : Wh_i = Wr_j$, $\exists_t = 1..n, z = 1..m : Wh_i = WHdd_t, Wr_j = WRam_z, \{Vhdd_t, Vram_z\} \in Ehr$;
      2.2.2 для пары $G_{ram}, G_{net} - \forall_i = 1..m, j = 1..k : Wr_i = Wn_j$, $\exists_t = 1..m$, $z = 1..k : Wr_i = WRam_t, Wn_j = WNet_z, \{Vram_t, Vnet_z\} \in Ern$;
      2.2.3 для пары $G_{net}, G_{hdd} - \forall_i = 1..k$, $j = 1..n : Wn_i = Wh_i$, $\exists_t = 1..l$, $z = 1..n : Wn_i = WNet_t, Wh_j = WHdd_z, \{Vnet_t, Vhdd_z\} \in Enh$.

Суть описанных выше алгоритмических операций заключается в сравнении значений элементов множеств *Wh*, *Wr*, *Wn*, представляющих собою набор вершин, и формировании следующих множеств *Ehh*, *Err*, *Enn*, *Ehr*, *Ern*, *Enh*, представляющих собою набор ребер. В дальнейшем эту информацию используют для внутреннего постинцидентного аудита средства вычислительной техники.

Теперь рассмотрим временную асимптотическую сложность алгоритма по соответствующим пунктам списка с учетом того, что множества будут храниться в массивах.

1. В каждом из данных множеств преобразование будет иметь сложность $O(n), O(m), O(k)$ действий для $WHdd, WRam, WNet$ соответственно:
   - 1.1 операции создания и копирования на массиве занимают $O(n), O(m), O(k)$;
   - 1.2 существующие алгоритмы сортировок имеют временную сложность $O\ (t \log(t))$, т.е. в нашем случае займут $O(n \log(n)), O(m \log(m)), O(k\log(k))$.
2. В данном пункте действия в каждом из подпунктов аналогичны тому, который описан в первом (2.1.1 и 2.2.1), для остальных приведены только итоговые результаты.
   - 2.1 Поиск ребер в каждом из подграфов:
     - 2.1.1 алгоритм поиска совпадающих значений будет иметь сложность $O(n\log(n))$, и поиска в исходном множестве имеет такую же сложность $O(n\log(n))$, и внесение результата занимает $O(n \log(n))$ и $O(n\log(n))$;
     - 2.1.2 $O(m \log(m))$ и $O(m\log(m))$;
     - 2.1.3 $O(k\log(k))$ и $O(k\log(k))$.
   - 2.2 Поиск ребер между подграфами:
     - 2.2.1 Поиск совпадающих значений между подграфами занимает $O(t\log(z))$, где t – количество элементов во множестве из которого проводим поиск, а z – количество элементов во множестве по которому проводим поиск. Поиск в исходных множествах будет занимать $O(t\log(t))$. Для данных пар подграфов это займет $O(n\log(m))$ и $O(n\log(n))$, $O(m\log(m))$;
     - 2.2.2 $O(m\log(k))$ и $O(m\log(m))$, $O(k\log(k))$;
     - 2.2.3 $O(k\log(n))$ и $O(k\log(k))$, $O(k\log(k)), O(n\log(n))$.

Подводя итог, просуммируем получившиеся результаты $O(n) + O(m) + O(k) + O(n) + O(m) + O(k) + O(n\log(n)) + O(m\log(m)) + O(k\log(k)) + 2O(n\log(n)) + 2(m\log(m)) + 2O(k\log(k)) + O(n\log(m)) + O(n\log(n)) + O(m\log(m)) + O(m\log(k)) + O(m\log(m)) + O(k\log(k)) + O(k\log(n)) + O(k\log(k)) + O(k\log(k)) + O(n\log(n))$, данное значение можно сократить до $O(n) + O(m) + O(k) + O(n\log(n)) + O(m\log(m)) + O(k\log(k)) + O(n\log(m)) + O(m\log(k)) + O(k\log(n))$, и в итоге получаем сложность, равную $O(n\log(n)) + O(m\log(m)) + O(k\log(k)) + O(n\log(m)) + O(m\log(k)) + O(k\log(n))$. Асимптотическая сложность внесения найденных ребер в отдельное множество не меняется в зависимости от алгоритма поиска ребер и будет составлять $O(|E|)$, где $|E|$ – количество найденных ребер.

Для апробации полученных результатов был проведен эксперимент. Цели эксперимента заключались в определении временных затрат на проведение постинцидентного внутреннего аудита средств вычислительной техники.

Эксперимент состоял в следующем:

1. с отобранного средства вычислительной техники снимались дампы памяти;
2. полученные дампы памяти разбирали на атрибуты и их значения;
3. атрибуты и их значения записывались в нереляционную базу данных;
4. применялась описанная выше методика, и строился граф;
5. при различных перестроениях графа определялись временные затраты, а также анализировалось повышение информативности инцидента при его визуализациях.

Описание эксперимента

В качестве объектов исследования был создан экспериментальный стенд, состоящий из компьютеров (10 штук) следующей конфигурации:

Процессор	Intel Core i7 2600K 3.4Ghz
Оперативная память	3 ГБ
Жесткий диск	320 ГБ

Каждый из этих компьютеров выступал в качестве постинцидентного средства вычислительной техники. Все они исследовались независимо друг от друга. Также была разработана программная реализация метода, которая выступала в качестве инструмента исследования. Она обрабатывает полученные данные (дампы) жесткого диска, оперативной памяти, сети с постинцидентного средства вычислительной техники, загружает в нереляционную (NoSQL) СУБД атрибуты и их значения, применяет разработанную методику для построения графа в NoSQL СУБД. Пример визуализации части графа представлен на рис. 2. Временные затраты на обработку и визуализацию экспериментов представлены в табл. 2.

На основе полученных результатов из табл. 2 были построены график отношения импорта данных к количеству вершин (рис. 3) и графики зависимости времени перестройки графа к количеству добавляемых связей (рис. 4–6).

На основании рис. 4 можно сделать вывод о том, что большая часть времени затрачивается на перестройку графа, а количество добавляемых связей не оказывает большого влияния на систему. Рис. 5, 6

иллюстрируют нагрузку на систему, выраженную во времени перестройки графа и зависящую от количества добавляемых связей. Но при этом максимальное время с учетом перестройки графа не превышает 6000 мс. Из всего этого можно сделать вывод о том, что время, затрачиваемое на анализ с помощью методики, значительно меньше, чем время сбора информации.

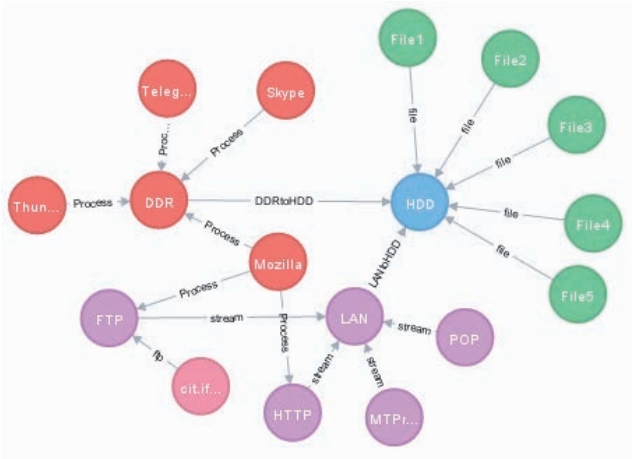

Рис. 2. Пример визуализации части графа на основе данных (атрибутов и их значений) с постинцидентного средства вычислительной техники

Время импорта данных, мс	Количество вершин	sid		handles		connections	
		строк	мс	строк	мс	строк	мс
146746	33906	8696	757	14047	1374	68	444
140095	37884	634	164	17367	2300	17	50
165791	49808	820	350	22254	2468	96	71
216069	60727	2016	774	27234	4116	47	179
100833	24995	463	69	11035	1134	5	48
189028	55537	968	218	25875	2882	43	195
101563	28621	600	144	12853	953	11	67
97046	27741	0	61	12253	1084	2	34
185788	46718	699	379	21854	2593	13	176
154048	37025	618	312	16975	1584	11	86

Таблица 2. Временные затраты на обработку и визуализацию графа

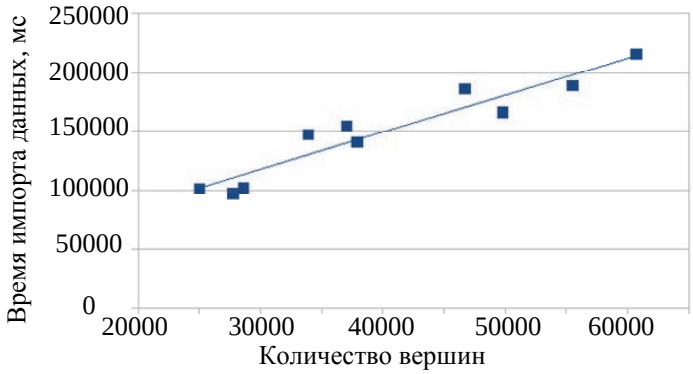

Рис. 3. Отношение времени импорта данных к количеству вершин

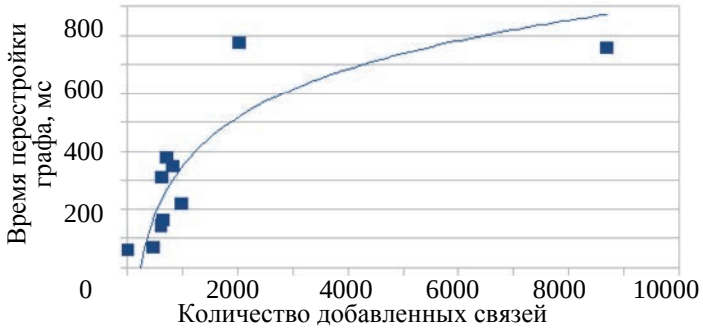

Рис. 4. Зависимость времени перестройки графа от количества добавляемых связей

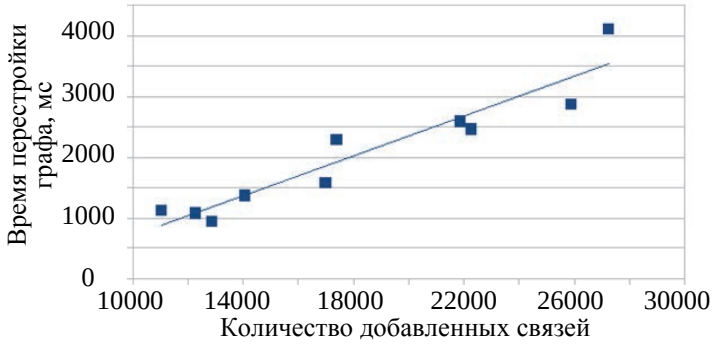

Рис. 5. Зависимость времени повторной перестройки графа от количества добавляемых связей

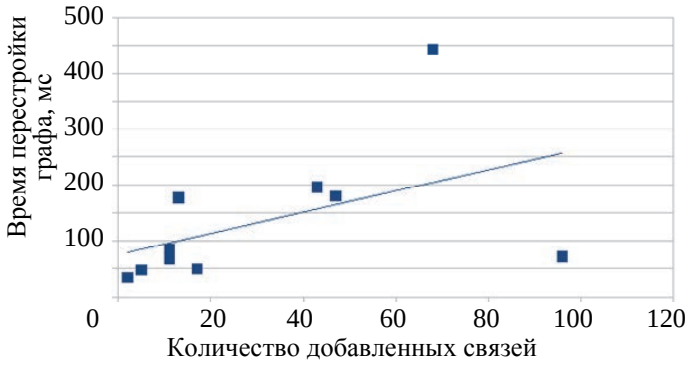

Рис. 6. Зависимость времени третьей перестройки графа от количества добавляемых связей

Заключение

Метод проведения постинцидентного внутреннего аудита средств вычислительной техники на основе графов позволяет устанавливать взаимосвязи между данными (дампами) жесткого диска, оперативной памяти, сети. Информативность результатов расследования инцидента информационной безопасности улучшилась за счет увеличения количества обрабатываемой информации и применения фильтрации по критериям в нереляционной системе управления базой данных. Также снизилась вероятность ошибки аудита данных (дампов) с постинцидентного средства вычислительной техники за счет автоматизации установления взаимосвязей в программной реализации. Представленный метод показал свою применимость в визуализации инцидента информационной безопасности. С применением определенных критериев к базе данных можно перестроить граф так, чтобы получить наглядную визуализацию инцидента информационной безопасности.

Достоинством представленного метода является то, что он позволяет снизить временные затраты на проведение внутреннего аудита средств вычислительной техники, повысить точность и информативность такого аудита. Данный метод имеет потенциал к развитию и может применяться с иными компонентами в задачах идентификации пользователей и задачах компьютерной криминалистики.

References

1. Derov E. *Uchityvaya bystroe razvitie i rost populyarnosti tekhnologii Big Data, est' prichina zadumat'sya o tselesoobraznosti ikh primeneniya pri rassledovanii intsidentov IB*. Available at: http://kabest.ru/press/news/754/index.php?print=Y (accessed 15.04.2016).

2. Carrier B. *File System Forensic Analysis*. Addison Wesley, 2005, 600 p.
3. Ligh M.H., Case A., Levy J., Walters A. *The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory*. Wiley, 2014, 912 p.
4. Davidoff S., Ham J. *Network Forensics: Tracking Hackers through Cyberspace*. Prentice Hall, 2012, 576 p.
5. Bessonova E.E., Zikratov I.A., Roskov V.Yu. Analysis of Internet user identification methods. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2012, no. 6(82), pp. 128–129.
6. Bessonova E.E., Zikratov I.A., Kolesnikov Yu.L., Roskov V.Yu. Internet user identification method. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2012, no. 3(79), pp. 133–137.
7. Limon G.G. Forensic physical memory analysis: an overview of tools and techniques. In: *TKK T-110.5290 Seminar on Network Security*. Helsinki, Finland, 2007, pp. 305–320.
8. Carrier B.D., Grand J. A hardware-based memory acquisition procedure for digital investigations. *Digital Investigation*, 2004, vol. 1, no. 1, pp. 50–60. doi: 10.1016/j.diin.2003.12.001.
9. Wang W. A graph oriented approach for network forensic analysis. Graduate Theses and Dissertations. Iowa State University, 2010, 122 p..
10. Jajodia S., Noel S., O'Berry B. Topological analysis of network attack vulnerability / In: Managing Cyber Threats: Issues, Approaches and Challenges. Springer-Verlag, 2005. P. 248-266.
11. Vicknair C., Nan X., Macias M., Chen Y., Zhao Z., Wilkins D. A comparison of a graph database and a relational database: a data provenance perspective. *Proc. 48th Annual South-East Regional Conf., ACM SE'10*. Oxford, USA, 2010, art. 42. doi: 10.1145/1900008.1900067
12. Tanenbaum A.S. *Structured Computer Organization*. 6th ed. Pearson, 2012, 800 p.
13. Yurin I.V., Pantyukhin I.S. Testing the hypothesis of creating a digital polygraph based on video and audio data. *Vestnik Gosudarstvennogo Universiteta Morskogo i Rechnogo Flota imeni Admirala S.O. Makarova*, 2015, no. 3(31), pp. 202–209.
14. Khoroshevskii V.G. *Arkhitektura Vychislitel'nykh Sistem* [Architecture of Computer Systems]. Moscow, MGTU im. Baumana Publ., 2005, 510 p.
15. Harary F. *Graph Theory*. Addison-Wesley, 1969.
16. Tutte W.T. *Graph Theory as I Have Known It*. Oxford University Press, 2001, 360 p.
17. Christofides N. *Graph Theory: An Algorithmic Approach*. NY, Acadimic, 1975.
18. Bondy J.A., Murty U.S.R. *Graph Theory with Applications*. NY-Amsterdam-Oxford, North - Holland, 1976, 268 p.

Пантюхин Игорь Сергеевич – тьютор, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, zevall@cit.ifmo.ru

Зикратов Игорь Алексеевич – доктор технических наук, профессор, заведующий кафедрой, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, zikratov@cit.ifmo.ru

Левина Алла Борисовна – кандидат технических наук, доцент, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, levina@cit.ifmo.ru

Igor S. Pantiukhin – tutor, ITMO University, Saint Petersburg, 197101, Russian Federation, zevall@cit.ifmo.ru

Igor A. Zikratov – D.Sc., Professor, Head of Chair, ITMO University, Saint Petersburg, 197101, Russian Federation, zikratov@cit.ifmo.ru

Alla B. Levina – PhD, Associate professor, Associate professor, ITMO University, Saint Petersburg, 197101, Russian Federation, levina@cit.ifmo.ru