3. Полежаев П.Н., Ушаков Ю.А., Шухман А.Е. Система управления ресурсами для высокопроизводительных вычислений, основанная на использовании программно-конфигурируемой сети // "Системы управления и информационные технологии: научно-технический журнал", 2013. - №4(54). - С. 65-69.

Д.А. Рыбаков

ЗАЩИТА ДОКУМЕНТОВ ПО ПРИНЦИПУ “ЗАПРЕЩЕНО ВСЁ, ЧТО НЕ РАЗРЕШЕНО”

(EPAM Systems)

В 1990х годах сложность систем таких как MS-DOS или Linux была невысокой, и поэтому было под силу разобраться одному специалисту и полностью взять её под контроль. Пара сотен системных функций, небольшие объемы данных, несколько программ вполне укладывались в голове одного человека и было совершенно понятно, что делает система и с чем взаимодействует. На данный момент сложность информационных систем возросла до такого уровня, что трудно достоверно сказать, чем занимается операционная система помимо основных своих обязанностей. Плюс большое количество программ, которые так и лезут в компьютер со всех уголков интернета. Эти тенденции делают жизнь простого пользователя совершенно непредсказуемой, когда дело касается сохранения конфиденциальной информации.

Несмотря на такое возрастающее разнообразие, фирма Microsoft неуклонно реализует принцип защиты “разрешено всё, что не запрещается”, а запрещается не так уж и много, то есть по большому счету информация остается слабозащищённой. Такой подход имеет свои положительные стороны, так как система всегда выполняет множество своих функций, что сильно уменьшает время настройки и развертывания, что выгодно с коммерческой точки зрения и не требует специальных навыков от пользователя. Но за это приходится платить повышенными рисками и необходимостью иметь антивирусы, которые потребляют ощутимую часть ресурсов. Задача администратора системы при таком подходе – затыкать бреши в защите по мере их обнаружения.

Другое отношение к защите имеет операционная система Unix. В ней реализуется принцип “запрещено всё, что не разрешено”. То есть по умолчанию большинство возможностей закрыты, и администратор дает разрешения и открывает доступы к различным службам, файлам, устройствам по мере необходимости. Такой подход увеличивает время развертывания и настройки, зато сокращает последующие расходы на сопровождение и делает систему более предсказуемой.

Не отрицая ни первого, ни второго подхода автор предлагает гибридный подход к защите файлов. Для значимой информации предлагается реализовать жесткий принцип “запрещено всё, что не разрешено”. А для остального боль-

шого количества файлов остается свободный подход “разрешено всё, что не запрещается”. Таким образом можно совместить достоинства обоих путей.

Еще более жесткий контроль заключается в том, чтобы регулировать, какие программы могут читать какие виды файлов. Например, файлы с расширением .doc могут читаться и записываться только приложениями MicrosoftOffice. Нет необходимости давать всем программам полный доступ к этому виду файлов. Так же нет необходимости давать офисным приложениям какой-либо доступ к файлам, например, с расширением .sys, которые содержат коды драйверов системы. Также программы MicrosoftOffice вполне могут читать .dll файлы, но не модифицировать их. То есть предлагается реализовать привязку типов файлов и видов операций к приложениям, которые их обслуживают.

У критически настроенного читателя могут найтись много возражений против такого подхода и это естественно. Кончено, полная абсолютная защита невозможна, тем не менее считаю, что дополнительные препятствия вполне могут остановить большое количество автоматических шпионских программ и сделать взлом более дорогостоящим, чем стоимость самой информации.Естественно, что предполагается контролируемое расширение возможностей при необходимости. Например, если будет установлен другой офисный пакет, то он тоже потребует разрешения на чтение .doc файлов.

Близкими конкурентами данного подхода являются системы EncFs [1], AppArmor [2], InfoWatch [3], но они во многом отличаются от задуманной архитектуры и не выполняют всех задуманных функций. Близкой также является находящаяся в процессе разработки система Monadnock, основанная на Data modification kit от компании OSR [4]. Познакомится с продуктами можно по приведенным ссылкам. По сути предложенный вид защиты является новым.

Рассмотрим подробнее доступ к .doc файлам. Приложения можно условно разделить на три класса. Первый класс – родные доверенные приложения, которые непосредственно создают и читают эти файлы. Второй класс – вспомогательные, например, почтовые клиенты, файловые менеджеры, которые могут копировать, пересылать файлы не интересуясь содержимым. Третий класс – остальные неизвестные программы, которым запрещено что-либо делать.

Теперь предположим, что наша система следующим образом обрабатывает запросы. Для доверенных программ разрешается полный набор операций, как это происходит в нормальной привычной системе. Тут ничего нового нет. А для вспомогательных программ система дает читать .doc файл, но при этом она сначала зашифровывает данные.Таким образом вспомогательная программа пересылает зашифрованные данные, и по дороге они не будит раскрыты третьим лицам. Что в общем достаточно для таких программ, так как они не занимаются модификацией данных, да и сами данные для них не интересны. Остается только расшифровывать данные при получении от такой программы, чтобы можно было передать доверенным приложениям. Теперь допустим, что в организации на все компьютеры установлена данная система, тогда она будет автоматически производить шифрование и расшифровку, и поэтому простые пользователи ничего не заметят.А если файл уйдет за пределы организации, то он останется за-

шифрован и бесполезен для третьих лиц, так как у них нет ни ключей, ни алгоритмов расшифровки. Таким образом, реализуя несложную схему, мы получаем замкнутую систему внутри организации, из которой не утекает информация в виде файлов. Это подобно введению своего крипто-языка в компании друзей, когда только друзья понимают, о чем они говорят. Таким образом можно беспрепятственно пользоваться сетевыми службами типа Dropbox, YandexDisk и подобными и быть уверенным, что файлы при выходе за пределы организации будут шифроваться и при поступлении обратно расшифровываться.

Что же получают неизвестные программы? В зависимости от конфигурации они могут либо вообще ничего не получать, либо получать какие-то случайные данные, чтобы запутать злоумышленников.

Возникает резонный вопрос, каким образом можно реализовать необходимый функционал? Так как все запросы к файлам проходят через систему драйверов, то имеет смысл внедряться на этот уровень программного обеспечения и контролировать все операции с этого уровня [5]. В OCMicrosoftWindows, существует уровень FileSystemFilterDrivers [6], который служит для перехвата обращений к файлам. В момент обработки запросов на открытие файлов IRP_MJ_CREATE можно получить с помощью процедуры ZwQueryInformationProcessимя процесса, который осуществляет запрос на открытие файла. Далее в этом же обработчике с помощью процедуры IoCancelFileOpen можно отменить открытие файла и передать статус STATUS_ACCESS_DENIED в уровень приложения, блокируя открытие.

Намного более сложной является операция шифрования файлов. Это связано с тем, что в драйвер поступают множество запросов для получения длины файла и чтения, записи, блокировки произвольных фрагментов. Поэтому, если использовать алгоритмы шифрования, которые изменяют размер файла и меняют байты и биты местами, то возникают множество трудностей, особенно с большими файлами, которые не помещаются в оперативную память и cmemorymappedfiles, с которыми алгоритм работы отличается от стандартного.

Для демонстрации автором бы создан простейший FilesystemFilterдрайвер, который анализирует имя файла и имя приложения, выполняющего открытие файла и принимает решение давать ли право на открытие или нет. Это подтверждает реализуемость данного подхода. Более сложные манипуляции можно сделать, если модифицировать исходные коды TrueCrypt или EncFs.

Автор выражает благодарность А.Мидрину, Е.Яфаркину за поддержку данного проекта.

Литература

1. EncFScryptographicfilesystem [Электронный ресурс] / Сайт Wikipedia. Режимдоступа: http://en.wikipedia.org/wiki/EncFS

2. Application ArmorLinux kernel security module [Электронныйресурс]/ СайтWikipedia. Режимдоступа: http://en.wikipedia.org/wiki/AppArmor

3. Сайт компании InfoWatch[Электронный ресурс]/ Режим доступа: http://www.infowatch.ru

4. The Next Generation Encryption Solution Kit[Электронныйресурс]/ Сайт-компанииOSR. Режимдоступа: http://www.osr.com/monadnock/

5. WindowsDriverKit [Электронныйресурс]/ СайтWikipedia. Режим доступа: http://en.wikipedia.org/wiki/Windows_Driver_Kit

6. File System Filter Drivers[Электронныйресурс]/СайткомпанииMicrosoft. Режим доступа: https://msdn.microsoft.com/en-us/library/windows/hardware/ff540382(v=vs.85).aspx

В.П. Цветов

ОБ ОДНОМ СИНТАКСИЧЕСКОМ АЛГОРИТМЕ НА ГРАФАХ

(Самарский государственный университет)

Алгоритмы на графах представляют интерес для различных прикладных направлений, в частности, они могут быть полезны при проектировании информационных систем или построении систем безопасности. Существующие алгоритмы используют различные подходы, опирающиеся на специальные представления и структуры данных, обширная библиография по которым представлена, например в [1].

В статье рассматривается алгоритм нахождения оптимальных маршрутов на реберно-размеченных графах. Результат не претендует на практическую новизну, но демонстрирует возможный абстрактно алгебраический метод решения задач, связанных с построением степеней элементов ограниченной сверху структурно упорядоченной полугруппы [2] в матричном представлении.

Введем следующие обозначения.

$\mathbb{N}$-множество натуральных чисел;

$1..n := \{1,2,\dots,n\} \subset \mathbb{N}$;

$\mathcal{A} := \{\alpha_1,\alpha_2,\dots,\alpha_n\}$- алфавит мощности n;

$\mathbb{W}_{\mathcal{A}}^{k}$- множество слов длины k над алфавитом $\mathcal{A}$;

$\mathbb{W}_{\mathcal{A}} := \bigcup_{k=0}^{\infty} \mathbb{W}_{\mathcal{A}}^{k}$- множество слов над алфавитом $\mathcal{A}$;

$\langle \mathbb{W}_{\mathcal{A}}, (\bullet) \rangle$- моноид слов над алфавитом $\mathcal{A}$ с операцией сцепления;

$\mathbb{R}$- множество вещественных чисел;

$\mathbb{R}_{+}$- множество неотрицательных вещественных чисел;

$[0,1] \subset \mathbb{R}$- замкнутый интервал вещественных чисел от 0 до 1;

$\langle \mathbb{R}, (\cdot,+) \rangle$- поле вещественных чисел;

$\langle \mathbb{R}, (\wedge,\vee,) \rangle$- решетка вещественных чисел, где $\xi_1 \wedge \xi_2 := \min(\xi_1,\xi_2)$, $\xi_1 \vee \xi_2 := \max(\xi_1,\xi_2)$;

$\mathbb{R}^{\infty} := \mathbb{R} \cup \{\iota\}$- расширенное множество вещественных чисел;

$\langle \mathbb{R}^{\infty}, (\wedge,\vee,\cdot,+) \rangle$- расширенная алгебра вещественных чисел, где $\xi \in \mathbb{R}$ и $\xi \wedge \iota = \iota \wedge \xi := \xi$, $\xi \vee \iota = \iota \vee \xi = \xi \cdot \iota = \iota \cdot \xi = \xi + \iota = \iota + \xi = \iota \cdot \iota = \iota + \iota := \iota$;

$\langle \mathbb{R}_{+}^{\infty}, (\wedge,\vee,\cdot,+) \rangle$- подалгебра $\langle \mathbb{R}^{\infty}, (\wedge,\vee,\cdot,+) \rangle$ с носителем $\mathbb{R}_{+}^{\infty}$;