УДК 004.056.53
DOI: 10.12737/17148

А.В. Козачок, О.М. Голембиовская, Л.М.Туан

ПРОТОТИП СИСТЕМЫ КОНТРОЛИРУЕМОГО РАЗГРАНИЧЕНИЯ ДОСТУПА К ФАЙЛАМ ДОКУМЕНТАЛЬНЫХ ФОРМАТОВ

Рассмотрены вопросы разработки прототипа системы контролируемого разграничения доступа к файлам документальных форматов. Приведено описание процесса функционирования системы контролируемого разграничения доступак файлам документальных форматов, отличающейся применением процедуры неразличимой обфускации. Представлены научно-технические предложения по реализации системы контролируемого разграничения доступа.

Ключевые слова: защита информации, система разграничения доступа, обфускация, изолированная программная среда, программный код.

A.V. Kozachok, O.M. Golembiovskaya, L.M. Tuan

PROTOTYPE OF ACCESS CONTROLLED DIFFERENTIATION SYSTEM TO FILES OF DOCUMENTATION FORMATS

The safety problem with information circulating in corporate information-computer nets is urgent under conditions of present-day information society. The authors have developed a generalized functional model of the process of controlled access differentiation. At the same time come forward users identified by accounts as access subjects in the model and files of documentation formats are objects. Rules for the differentiation of a subject access to objects are specified as a matrix of powers taking into account marks of confidentiality. A distinguishing feature consists in that a container storing data is protected on basis of the method of indistinguishable obfuscation. The model developed allows storing data in a uniformed kind and ensuring a single method for an access to them. For safe storing is used a format of the protected container where information is stored in an obfuscated form. A container represents an executable file having a number of preset properties and functions allowing unambiguously the user identification, differentiation of an access to data (rights: to read, write, and assignation), assurance of the security for a confidence of the document implemented. The container format ensures its safe storing and transmission through a network.

Key words: information security, access differentiation system, obfuscation, isolated program environment, program code.

В настоящее время вопрос обеспечения безопасности информации, циркулирующей в корпоративных информационно-вычислительных сетях, стоит остро. Это связано не только с широким спектром угроз безопасности, но и с ограниченными возможностями имеющихся в наличии средств защиты, которые не позволяют обеспечить защищенность автоматизированных систем от ряда угроз на должном уровне.

Целью проводимого исследования является построение комплекса моделей и алгоритмов процесса контролируемого разграничения доступа к файлам документальных форматов, позволяющего осуществить защиту от несанкционированного доступа к информации за счет применения неразличимой обфускации программного кода [1].

Исходя из изложенного разработана обобщенная функциональная модель процесса контролируемого разграничения доступа. При этом субъектами доступа в модели выступают пользователи, идентифицируемые учетными записями, а объектами являются файлы документальных форматов. Правила разграничения доступа субъектов к объектам задаются в виде матрицы полномочий, учитывающей метки конфиденциальности.

В основу предлагаемой модели положен подход, схожий с моделью системы военных сообщений [2], так как используется понятие контейнера для обработки структурированных данных. Отличительная особенность заключается в том, что контейнер является защищенным на основе метода неразличимой обфускации [3].

Разработанная модель позволяет хранить данные в унифицированном виде и обеспечивает единый метод доступа к данным всех типов. Для безопасного хранения используется формат защищенного контейнера, в котором данные хранятся в обфусцированном виде. Контейнер пред-

ставляет собой исполняемый файл, обладающий рядом заданных свойств и функций, позволяющих однозначно идентифицировать пользователя, разграничивать доступ к данным (права: читать, писать, передать права), обеспечивать защиту конфиденциальности внедренного документа. Формат контейнера обеспечивает его безопасное хранение и передачу по сети.

На основе разработанной модели предлагается прототип системы контролируемого разграничения доступа к файлам документальных форматов. На рис. 1 представлена обобщенная архитектура прототипа системы контролируемого разграничения доступа.

Алгоритм работы с системой предполагает осуществление следующих действий:

1. Пользователь осуществляет попытку доступа к документу, хранящемуся в формате защищенного контейнера. Подсистема контроля и разграничения доступа идентифицирует пользователя и проверяет его права доступа к данному документу.

2. В случае успешной идентификации пользователя и проверки прав доступа документ извлекается из защищенного контейнера в подсистему изолированной программной среды. В противном случае доступ запрещается.

3. Извлекаемый документ помещается в изолированную программную среду для дальнейшей работы с ним пользователя. По окончании работы с документом выполняется процедура назначения прав доступа к нему. Затем документ с правами доступа инкапсулируется в формат защищенного контейнера.

4. После внедрения документа в формат контейнера осуществляется процедура неразличимой обфускации над данным контейнером.

5. Затем осуществляется проверка корректности контейнера, заключающаяся в проверке соответствия спецификации исполняемого файла и назначенных прав доступа.

6. Полученный обфусцированный контейнер готов к передаче и дальнейшей работе с ним.

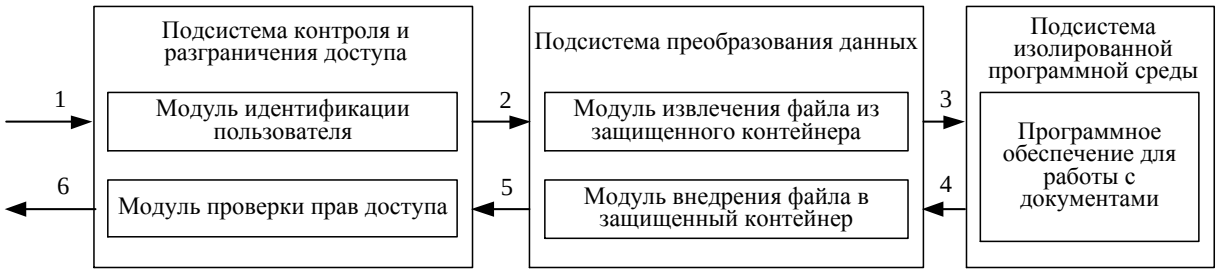

Рис. 1. Обобщенная архитектура прототипа системы контролируемого разграничения доступа к файлам документальных форматов

В состав системы контролируемого разграничения доступа к файлам документальных форматов входят следующие подсистемы:

– контроля и разграничения доступа;

– преобразования данных;

– изолированной программной среды.

Основными компонентами подсистемы контроля и разграничения доступа являются модуль идентификации пользователя и модуль проверки прав доступа. Модуль идентификации пользователя выполняет идентификацию пользователей, аутентифицированных в операционной системе, за счет информации о текущем сеансе работы пользователя при осуществлении доступа к обфусцированному контейнеру. Модуль проверки прав доступа реализует проверку прав доступа пользователей к обфусцированному контейнеру. Матрица полномочий доступа хранится в формате защищенного контейнера в качестве дополнительной секции исполняемого файла.

Подсистема преобразования данных состоит из модулей внедрения файла в формат защищенного контейнера и извлечения файла из защищенного контейнера. Данная подсистема реализует следующие функции:

– подготовка шаблона исполняемого файла, соответствующего формату защищенного контейнера;

– внедрение файла документального формата и прав доступа к нему в шаблон исполняемого файла;

– извлечение файла из защищенного контейнера в изолированную программную среду;

– применение неразличимой обфускации к шаблону исполняемого файла.

Целью функционирования подсистемы изолированной программной среды является обеспечение защиты от несанкционированного доступа к файлам документальных форматов в оперативной памяти и на носителях информации в процессе их обработки.

В качестве базовой операционной системы (ОС) для реализации прототипа системы контролируемого разграничения доступа к файлам документальных форматов была выбрана ОС семейства Linux (Ubuntu 14.04). Обусловлено это тем, что данные ОС имеют открытый исходный код и широко применяются в различных государственных организациях и ведомственных структурах.

В ОС семейства Linux реализованы два уровня привилегий выполнения кода: уровень ядра и уровень пользователя. Таким образом, подход к построению системы контролируемого разграничения доступа к файлам документальных форматов на основе применения неразличимой обфускации предполагает реализацию данной системы на уровне ядра. Достоинствами данного подхода являются более высокая производительность по сравнению с технологиями, использующими уровень пользователя, а также возможность реализации изолированной программной среды, обеспечивающей защиту от несанкционированного доступа. На рис. 2 представлено место системы контролируемого разграничения доступа к файлам документальных форматов в архитектуре ОС семейства Linux.

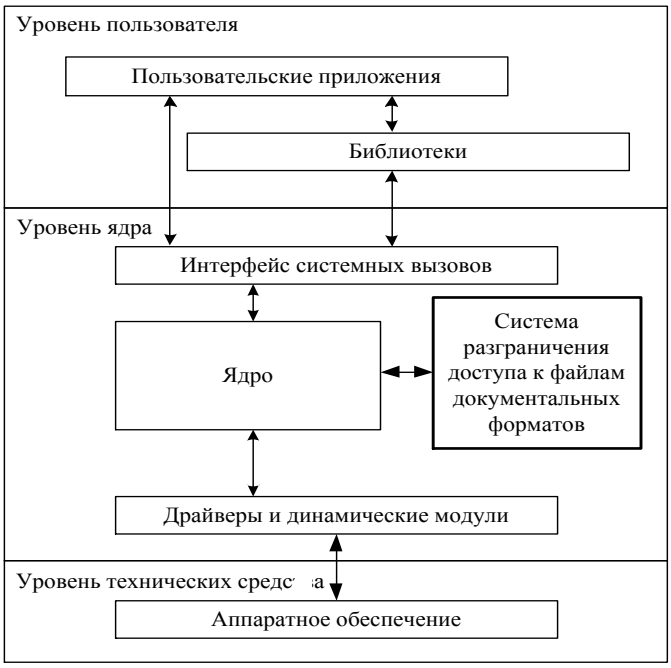

Рис. 2. Система контролируемого разграничения доступа к файлам документальных форматов в архитектуре ОС семейства Linux

На основе предложенной архитектуры была разработана структурная модель функционирования системы контролируемого разграничения доступа к файлам документальных форматов (рис. 3).

Необходимо отметить, что документ, извлеченный из обфусцированного контейнера, помещается в пространство изолированной программной среды. Таким образом, злоумышленник не может осуществить несанкционированный доступ к нему, поскольку документ не хранится в оперативной памяти, доступной для приложений уровня пользователя.

Согласно предложенной структурной модели функционирования системы контролируемого разграничения доступа к файлам документальных форматов, предполагаются следующие этапы:

1. Пользователь (user) осуществляет попытку получения доступа к документу в обфусцированном контейнере. Подсистема контроля и разграничения доступа (daemon) осуществляет идентификацию пользователя и проверку прав доступа. В случае разрешения доступа документ извлекается из защищенного контейнера и помещается в каталог «input_buf», который одновременно подключен к изолированной программной среде. Права доступа к этим каталогам обозначены на рис. 3.

2. Пользователь работает с документом в изолированной программной среде. По окончании работы пользователь назначает права доступа к документу. Затем документ сохраняется в каталог «output_buf», который является каталогом изолированной программной среды, но при этом подключен к базовой ОС. Пользователь не имеет прав на доступ к каталогу «output_buf», лишь подсистема контроля и разграничения доступа может получить доступ к данному каталогу.

3. При появлении документа в каталоге «output_buf» подсистема преобразования данных внедряет документ и права доступа в шаблон исполняемого файла, а затем применяется процедура неразличимой обфускации.

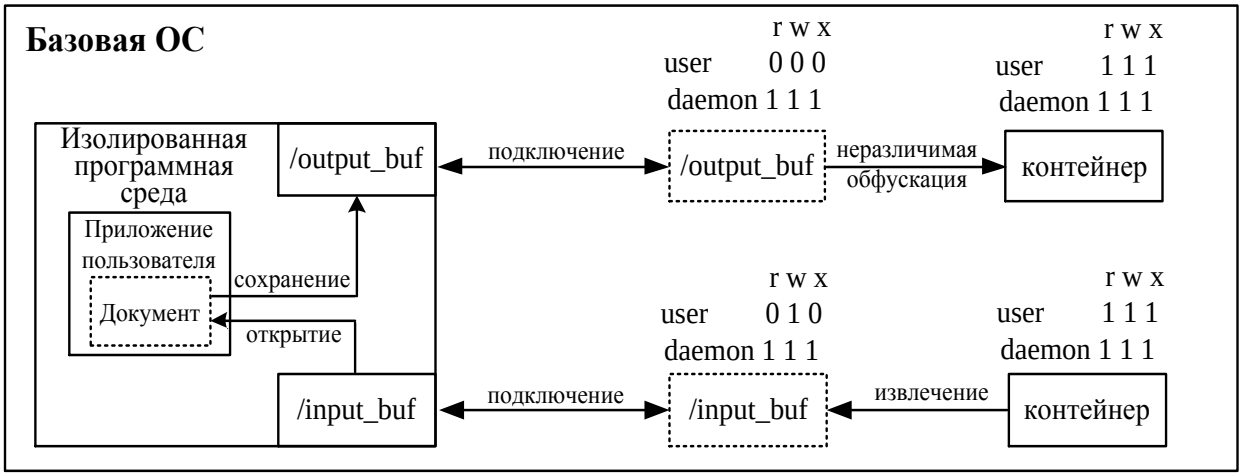

Рис. 3. Структурная модель функционирования системы контролируемого разграничения доступа к файлам документальных форматов

Для реализации изолированной программной среды был выбран подход на основе технологии Linux Containers (LXC). Технология LXC представляет собой систему виртуализации на уровне ОС, которая создает виртуальное окружение с собственным пространством процессов и сетевым стеком [4]. В LXC встроены механизмы контроля и ограничения доступа к ресурсам памяти и файловой системы. Для доступа к LXC предлагается использовать систему Docker (программное обеспечение для автоматизации развертывания и управления приложениями в среде виртуализации на уровне ОС). Она позволяет инкапсулировать приложение со всем его окружением и зависимыми библиотеками в контейнер, который может быть перенесен на любую ОС семейства Linux, с поддержкой механизма ядра cgroups. Docker имеет клиент-серверную архитектуру[4].

Клиентская часть позволяет из интерфейса командной строки управлять контейнерами. Серверная часть обеспечивает полную изоляцию запускаемых на узле контейнеров на уровне файловой системы (у каждого контейнера собственная корневая файловая система), на уровне процессов (процессы имеют доступ только к собственной файловой системе контейнера, а ресурсы разделены средствами LXC), на уровне сети (каждый контейнер имеет доступ только к привязанному к нему сетевому пространству имен и соответствующим виртуальным сетевым интерфейсам).

Docker-контейнер работает в пользовательском пространстве базовой ОС как изолированный процесс и использует ресурсы ядра совместно с другими контейнерами. При этом каждый Docker-контейнер представляет собой изолированную программную среду, которая является безопасной платформой для извлечения документов из обфусцированных контейнеров и работы с ними.

Реализация механизмов изоляции базируется на применении встроенных в яд-

ро ОС Linux штатных механизмов на основе пространств имен (namespaces) и групп управления (cgroups). Для создания контейнеров используются модули: libcontainer (namespaces и cgroups), lxc, Sstemd-nspawn, libvirt и другие механизмы безопасности [5]. Для формирования контейнера достаточно загрузить базовый образ окружения (dockerpullbase), после чего можно запускать в изолированном окружении произвольное приложение. Также в системе Docker реализованы групповая политика безопасности, ролевое управление доступом (SELinux, AppArmor) и минимизация возможностей ядра ОС, доступных контейнерам. На рис. 4 представлена архитектура системы изолированной программной среды Docker.

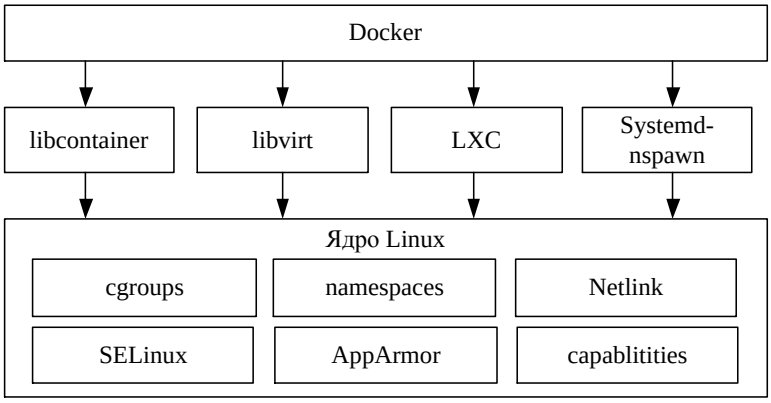

Рис. 4. Архитектура системы изолированной программной среды Docker

Применение системы Docker дает следующие основные преимущества [6]:

– запуск программного обеспечения для работы с документами, извлеченными из обфусцированного контейнера, в изолированной программной среде;

– защита конфиденциальности и целостности документов, извлеченных из обфусцированного контейнера;

– использование универсальных контейнеров для изоляции процессов от других процессов в базовой ОС;

– изоляция на уровне файловой системы (каждый процесс выполняется в отдельной корневой файловой системе).

Важным моментом программной реализации прототипа системы контролируемого разграничения доступа к данным является формирование шаблона исполняемого файла. Формат защищенного контейнера на основе шаблона исполняемого файла представлен на рис. 5. В рамках исследования в качестве исполняемого файла был выбран ELF–файл (Exucutable and Linkable Format), используемый во многих современных ОС семейства Linux [7].

Исполняемый файл, соответствующий спецификации ELF, состоит из ряда заголовков, содержащих служебные данные, предназначенные для информирования загрузчика ОС о структуре и размерах данных; секций файла, содержащих данные и машинный код, реализующий функциональное предназначение данной программы. Заголовок файла (.header) имеет фиксированное расположение в начале файла и содержит общее описание структуры файла и его основные характеристики, такие как: тип, версия формата, архитектура процессора, виртуальный адрес точки входа, размеры и смещения остальных частей файла [7].

Секции файла – это блоки данных, содержащие информацию о местоположении машинного кода и данных программы в файле, а также место расположения этих данных в виртуальном адресном пространстве и сами данные. Как видно из представленной структуры контейнера (рис. 5), секция .code содержит следующие основные функции, реализующие базовый функционал системы контроля и разграничения доступа:

– RunApp() – предназначена для загрузки и инициализации всех компонентов контейнера и системы контроля и разграничения доступа;

– UserID() – предназначена для получения идентификатора пользователя из информации о текущем сеансе работы пользователя при осуществлении доступа к обфусцированному контейнеру;

– CheckRights() – предназначена для проверки прав доступа пользователя к документу в обфусцированном контейнере в

соответствии с матрицей полномочий доступа, хранящейся вместе с документом;

– Extract() – предназначена для извлечения документа из обфусцированного контейнера.

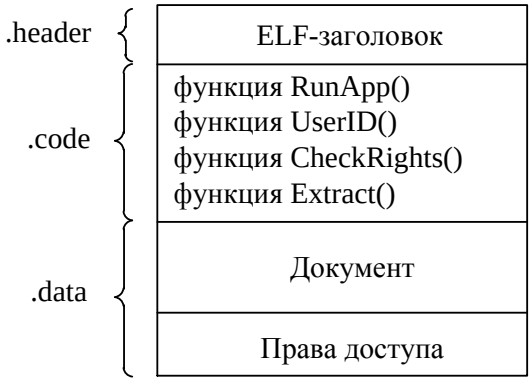

Рис. 5. Формат защищенного контейнера на основе шаблона исполняемого файла

Для открытия документа, находящегося в защищенном контейнере, пользователь запускает на исполнение обфусцированный контейнер. В случае предоставления доступа системой контроля и разграничения доступа осуществляется запуск приложения пользователя в изолированной программной среде с помощью системы Docker (рис. 6).

Секция .data предназначена для хранения документа при его инкапсуляции в формат защищенного контейнера, а также матрицы полномочий доступа, назначаемой пользователем по окончании работы с документом.

Рис. 6. Интерфейс приложения пользователя OpenOffice 4.0 в контейнере Docker

После запуска Docker–контейнера появляется окно приложения пользователя для работы с документом, извлеченным из обфусцированного контейнера. Далее пользователь может выбрать или открыть извлеченный документ или создать новый. Пример открытия существующего документа, извлеченного из обфусцированного контейнера в изолированную программную среду, представлен на рис. 7.

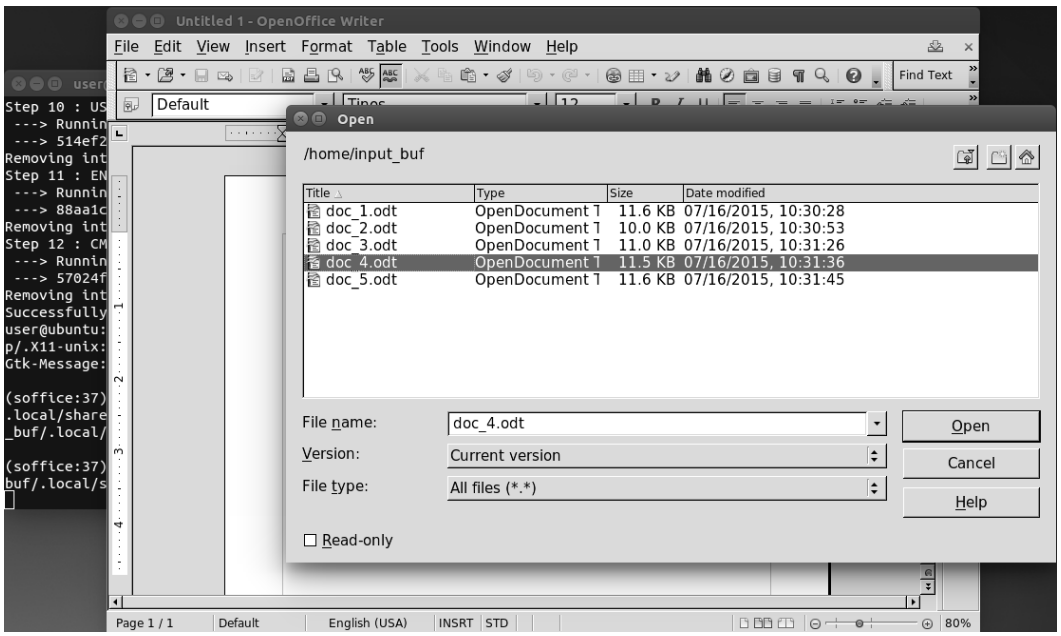

Рис. 7. Интерфейс приложения пользователя OpenOffice 4.0 при открытии существующих документов в изолированной программной среде Docker–контейнера

В соответствии со структурной моделью функционирования системы контролируемого разграничения доступа к файлам документальных форматов к Docker–контейнеру подключаются каталоги базовой ОС с соответствующими правами доступа.

По окончании работы с документом, в случае создания нового документа при наличии полномочий назначения прав доступа, пользователю необходимо заполнить матрицу полномочий доступа к нему.

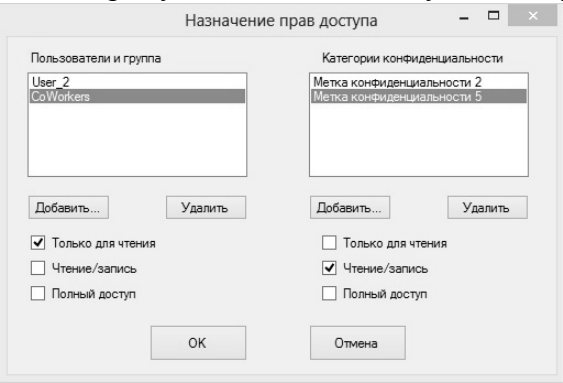

Рис. 8. Интерфейс прототипа окна назначения прав доступа

На рис. 8 показан интерфейс прототипа приложения для задания матрицы полномочий доступа. Как видно из представленного прототипа, данное приложение имеет возможность задавать правила разграничения доступа пользователей, комбинируя в себе элементы мандатного и дискреционного разграничения доступа. Основные особенности приложения:

– варианты ограничения прав (только чтение, чтение/запись и полный доступ);

– задание прав доступа для групп пользователей;

– задание прав доступа для отдельных пользователей;

– задание прав доступа по метке конфиденциальности.

В рамках исследования были рассмотрены следующие варианты внедрения матрицы полномочий в формат защищенного контейнера:

– расширение последней секции исполняемого файла;

– создание новой секции исполняемого файла.

Оба варианта имеют определенные достоинства и недостатки. Но внедрение на основе расширения последней секции может приводить к возникновению ошибок при исполнении кода. Поэтому в работе был выбран подход на основе создания новой секции, поскольку отсутствуют ограничения на объем внедряемой матрицы полномочий и файл при этом исполняется корректно.

Таким образом, исходя из представленного описания прототипа системы контролируемого разграничения доступа к файлам документальных форматов, можно сделать вывод, что применение системы контролируемого разграничения доступа к данным позволит добиться, с учетом ряда ограничений, предотвращения возможности несанкционированного доступа, а тем самым и возможности нарушения конфиденциальности и целостности обрабатываемой в автоматизированной системе информации.

СПИСОК ЛИТЕРАТУРЫ

1. Варновский, Н. П. Современное состояние исследований в области обфускации программ: определение стойкости обфускации / Н. П. Варновский, В. А. Захаров, Н. Н. Кузюрин [и др.] // Труды Института системного программирования РАН: электрон. журн. – Т. 26. – № 3. – С. 167–198.
2. Девянин, П.Н. Модели безопасности компьютерных систем : учеб.пособие для студентов высш. учеб. заведений / П.Н. Девянин. – М.: Академия, 2005. – С. 144.
3. Sahai, A. How to Use Indistinguishability Obfuscation: Deniable Encryption / A. Sahai, B. Waters // CRYPTO ePrint 2011. – Mode of access: https://eprint.iacr.org/2013/454.pdf.
4. Lessard, M. Introduction to linuxcontainer(lxc) and Docker / M. Lessard // RedHatConf. – http://people.redhat.com/mlessard/mtl/presentations/jan2014/LXC-Docker.pdf.
5. Petazzoni, J. Container's Anatomy / J.Petazzoni // Linuxcon. – http://events.linuxfoundation.org/sites/events/files/slides/Anatomy%20of%20a%20container.pdf.
6. Petazzoni, J. LXC, Docker and the future of software delivery / J. Petazzoni // Linuxcon. – http://events.linuxfoundation.org/sites/events/files/slides/lcna13_petazzoni.pdf.
7. Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2. – http://www.x86.org/ftp/manuals/tools/elf.pdf.

1. Varnovsky, N.P., Current state of investigations in field of program obfuscation: definition of obfuscation durability / N.P. Varnovsky, V.A. Zakharov, N.N. Kuzyurin [et alii] // *Proceedings of System Programming Institute of RAS:* digizine. Vol. 26. – No 3. – pp. 167-198.
2. Devyanin, P.N., Models for Computer System Safety: textbook for college students / P.N. Devyanin. – M.: Academy, 2005. – pp. 144.
3. Sahai, A. How to Use Indistinguishability Obfuscation: Deniable Encryption / A. Sahai, B. Waters // CRYPTO ePrint 2011. – Mode of access: https://eprint.iacr.org/2013/454.pdf.
4. Lessard, M. Introduction to linuxcontainer(lxc) and Docker / M. Lessard // RedHatConf. – http://people.redhat.com/mlessard/mtl/presentations/jan2014/LXC-Docker.pdf.
5. Petazzoni, J. Container's Anatomy / J.Petazzoni // Linuxcon. – http://events.linuxfoundation.org/sites/events/files/slides/Anatomy%20of%20a%20container.pdf.
6. Petazzoni, J. LXC, Docker and the future of software delivery / J. Petazzoni // Linuxcon. – http://events.linuxfoundation.org/sites/events/files/slides/lcna13_petazzoni.pdf.
7. Tool Interface Standard (TIS) Executable and Linking Format (ELF) Specification Version 1.2. – http://www.x86.org/ftp/manuals/tools/elf.pdf.

Материал поступил в редколлегию 2.09.15.
Рецензент: д.т.н., профессор Брянского государственного технического университета Ф.Ю. Лозбинев

Сведения об авторах:

Козачок Александр Васильевич, к.т.н, сотрудник Академии ФСО России, тел.: (4862) 54-99-33, e-mail: alex.totrin@gmail.com.
Голембиовская Оксана Михайловна, к.т.н, доцент, начальник отдела организации научно-исследовательской работы студентов, аспирантов и молодых ученых Брянского государственного технического университета, тел.: (4832) 58-82-06, e-mail: bryansk-tu@yandex.ru.
Лай Минь Туан, сотрудник Академии ФСО России тел.: (4862)54-99-33, e-mail: alex.totrin@gmail.com.

Kozachok Alexander Vasilievich, Can.Eng., colleague of the Academy of FSO of Russia, Phone: (4862)54-99-33, e-mail: alex.totrin@gmail.com.
Golembiovskaya Oksana Mikhailovna, Can.Eng., Assistant Prof., Head of the Dep. of Scientific-Research Work for students, post graduate students and young scientists, Phone: 4832) 58-82-06, e-mail: bryansk-tu@yandex.ru.
Lay Min Tuan, Colleague of the Academy of FSO of Russia, Phone: (4862)54-99-33, e-mail: alex.totrin@gmail.com.