УДК 002.56(075.32)
DOI: 10.25559/SITITO.15.201901.154-163

Масштабируемая архитектура комплексов обеспечения сетевой безопасности

О. Ю. Гузев[1], **И. В. Чижов**[2,3*]
[1] ОАО «ИнфоТеКС», г. Москва, Россия
[2] Московский государственный университет имени М.В. Ломоносова, г. Москва, Россия
[3] Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия
[*] ichizhov@cs.msu.ru

Аннотация

Как правило, сертифицированные средства обеспечения сетевой безопасности представляют собой узкоспециализированные комплексы с неизменной программно-аппаратной платформой. Главным недостатком такой архитектуры является невозможность прозрачного масштабирования устройств с ростом вычислительных возможностей сети. Кроме того, усложняется разработка и поддержка такого комплекса, так как аппаратная платформа быстро устаревает, что приводит к необходимости её замены, а, значит, и к доработке программных компонентов для поддержки нового оборудования. В работе описывается масштабируемая архитектура комплексов обеспечения сетевой безопасности, которая позволяет производителям упростить процесс обновления и разработки средств защиты информации. Главным свойством новой архитектуры является ориентированность на предоставление совокупности специализированных микро-сервисов. Она строится на принципах виртуализации сетевых функций и использует понятие унифицированной доверенной программно-аппаратной платформы. Каждая сетевая функция запускается на некоторой программно-аппаратной платформе, работающей под управлением операционной системы с гипервизором. Ясно, что в случае сертификации по требованиям безопасности конечных продуктов, необходимо обеспечить доверие к аппаратной платформе, операционной системе и гипервизору. Однако архитектура требует унификации программно-аппаратной платформы для всех сетевых функций. Это упрощает разработчикам поддержку конечных продуктов. Благодаря единой доверенной платформе архитектура позволяет упростить процедуры сертификации по требованиям безопасности информации в процессе поддержки и развития конечного продукта. Балансировка нагрузки и согласованность архитектуры обеспечивается средствами, реализующими технологию децентрализованных распределённых реестров (блокчейн).

Ключевые слова: виртуализация сетевых функций, программно-конфигурируемая сеть, балансировка нагрузки, информационная безопасность, сертификация, ПКС, NFV, SDN, OpenFlow, блокчейн.

Благодарности: исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта «Исследование протокола <>» № 18-29-03124.

Для цитирования: *Гузев О. Ю., Чижов И. В.* Масштабируемая архитектура комплексов обеспечения сетевой безопасности // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 1. С. 154-163. DOI: 10.25559/SITITO.15.201901.154-163

Scalable Architecture of Network Security Systems

O. Yu. Guzev[1], I. V. Chizhov[2,3*]
[1] JSC "InfoTeCS", Moscow, Russia
[2] Lomonosov Moscow State University, Moscow, Russia
[3] Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Moscow, Russia
[*] ichizhov@cs.msu.ru

Abstract

As a rule, certified means of ensuring network security are highly specialized complexes with a constant hardware and software platform. The main disadvantage of this architecture is the impossibility of transparent scaling of devices when the computing power of the network increases. In addition, the development and support of such a complex is complicated, as the hardware platform quickly becomes obsolete, which results in the necessity for replacing it, which means that the software components for the support of new equipment need to be improved.
The paper describes the scalable architecture of network security systems, allowing manufacturers to simplify the process of updating and developing information security tools. The main feature of the new architecture is the focus on the provision of a set of specialized micro-services. It is based on the principles of virtualization of network functions and it uses the concept of a unified trusted software and hardware platform. Each network function runs on a hardware and software platform running by the hypervisor operating system. It is clear that in the case of certification for the security requirements of the final products, it is necessary to ensure confidence in the hardware platform, operating system and hypervisor. However, the architecture requires unification of the software and hardware platform for all network functions. This makes it easier for developers to support end-products. Thanks to a single trusted platform, the architecture allows simplifying certification procedures for information security requirements in the process of supporting and developing the final product. Load balancing and architecture consistency are provided by means of implementing the technology of decentralized distributed registries (blockchain).

Keywords: network functions virtualization, software-configured network, load balancing, information security, certification, SDN, NFV, OpenFlow, blockchain.

Acknowledgements: The study was carried out with the financial support of the Russian Foundation for Basic Research in the framework of the research project "Protocol Research <>;" No. 18-29-03124.

For citation: Guzev O.Yu., Chizhov I.V. Scalable Architecture of Network Security Systems. *Sovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2019; 15(1):154-163. DOI: 10.25559/SITITO.15.201901.154-163

Введение

Сертифицированные средства обеспечения сетевой безопасности зачастую представляют собой узкоспециализированные комплексы с неизменной программно-аппаратной платформой. Главным недостатком такой архитектуры является невозможность прозрачного масштабирования устройств с ростом вычислительных возможностей сети. Кроме того, усложняется разработка и поддержка такого комплекса, так как аппаратная платформа быстро устаревает, что приводит к необходимости её замены, а, значит, и к доработке программных компонентов для поддержки нового оборудования.

В работе предлагается новая архитектура комплексов обеспечения сетевой безопасности, которая позволит производителям упростить процесс обновления и разработки средств защиты информации. Главным свойством новой архитектуры является ориентированность на предоставление совокупности специализированных микро-сервисов. В документе новая архитектура носит название «Scalable Architecture of Network Security Equipment and Tools (Масштабируемая архитектура устройств и инструментов сетевой безопасности)» (SANSET). Укажем базовые принципы и технологии, на основе которых предлагается строить новую архитектуру.

Первый принцип – использование виртуализации. Выделим базовое понятие сетевой функции. Концептуально под сетевой функцией будем понимать совокупность виртуальных машин (возможно контейнеров), запущенных на основе единого шаблона в среде некоторого гипервизора (в том числе и контейнерного). При этом шаблон сетевой виртуальной машины включает описания образа виртуальной машины, состава программного обеспечения, базовых настроек и необходимых для работы ресурсов. В задачи каждой виртуальной машины входит обработка сетевого трафика, поступающего на входные интерфейсы, и передача либо обработанного трафика, либо служебной информации в выходные интерфейсы. В некотором смысле все виртуальные машины, составляющие сетевую функцию, являются идентичными устройствами, выполняющими обработку сетевого трафика. Более подробно концепция виртуализации сетевых функций (NFV) описана в работах[1].

Поясним на примере сетевой функции «криптомаршрутизатор». Сетевая функция «криптомаршрутизатор» определяется образом виртуальной машины с ПО, реализующим функции криптографического маршрутизатора. Для корректной работы функции требуется набор криптографических ключей для связи с другими криптомаршрутизаторами сети. Кроме того, необходимо определить внешние сетевые адреса для связи других устройств с криптографическим маршрутизатором. Возможны также настройки сетевой функции, специфичные для конкретного программного обеспечения (ПО) криптографического маршрутизатора. Запущенная сетевая функция включает себя работающие виртуальные машины, созданные из привязанного к сетевой функции образа, а также контекста, состоящего из набора значений конкретных настроек сетевой функции. В частности, для криптографического маршрутизатора, каждая виртуальная машина инициализирована одним и тем же набором криптографических ключей. Виртуальные машины могут иметь локальные IP-адреса, через которые осуществляется управление ими, однако с точки зрения внешних устройств сети они представляют собой единое сетевое устройство с едиными сетевыми адресами, используемыми на всех уровнях модели ISO/OSI.

Второй принцип – унифицированная доверенная программно-аппаратная платформа. Каждая сетевая функция должна запускаться на некоторой программно-аппаратной платформе, работающей под управлением операционной системы с гипервизором. Ясно, что в случае сертификации по требованиям безопасности конечных продуктов, необходимо использовать доверенную аппаратную платформу, операционную систему и гипервизор. Однако для всех сетевых функций выбранная программно-аппаратная платформа должна быть единой. Что означает поддержку в продуктах не аппаратной платформы, а конкретной версии операционной системы и гипервизора.

Третий принцип – масштабирование как свойство архитектуры. Масштабирование, резервирование и балансировка нагрузки на узлы должны быть не отдельной сетевой функцией или отдельными устройствами, а являться свойством архитектуры. Сетевая функция – это набор виртуальных машин, обрабатывающих трафик. Распределение трафика на узлы сетевой функции – базовое свойство архитектуры. При таком подходе нет необходимости в применении специализированных технологий балансировки и резервирования. Масштабируемость как свойство системы означает прозрачное для пользователя автоматическое изменение количества узлов сетевой функции при изменении объёма обрабатываемого трафика. Добавление аппаратных модулей приводит к увеличению совокупной пропускной способности системы и не должно требовать изменения ПО виртуальных машин, составляющих сетевую функцию, для поддержки новых модулей.

Четвёртый принцип – универсальность аппаратной платформы. Каждая сетевая функция должна иметь возможность запускаться на любом аппаратном узле. Так на одном сервере могут соседствовать виртуальные машины различных сетевых функций, принадлежащих нескольким арендаторам. Изоляция этих виртуальных машин должна быть свойством системы.

Описание базовых компонентов архитектуры SANSET

В составе архитектуры выделяются следующие базовые логические элементы:

- SANSET-коммутатор;
- SANSET-узел;
- SANSET-контроллер;
- SANSET-оператор.

SANSET-коммутатор представляет собой программно-опреде-

[1] ETSI GS NFV 002. Network Functions Virtualization (NFV); Architectural Framework. v1.1.1. ETSI, Tech. Rep., October 2013. [Электронный ресурс]. URL: https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf (дата обращения: 21.12.2018).; Operational Opportunities and Challenges of SDN/NFV Programmable Infrastructure. ATIS-I-0000044, Tech. Rep. // ATIS, 2014. Pp. 45. [Электронный ресурс]. URL: https://www.atis.org/docstore/product.aspx?id=28143 (дата обращения: 21.12.2018).; Kreeger L., Dutt D., Narten T., Black D. Network virtualization NVE to NVA control protocol requirements, Internet Draft, Internet Engineering Task Force. April 24, 2014. [Электронный ресурс]. URL: http://tools.ietf.org/html/draft-ietf-nvo3-nve-nva-cp-req-02 (дата обращения: 21.12.2018).

ляемый коммутатор. Он отвечает за обеспечение сетевой связности сетевых функций с локальной и внешней сетями.

SANSET-коммутатор может быть построен на основе программного SDN-коммутатора Open vSwitch[2] с открытым исходным кодом. Этот программный коммутатор обеспечивает поддержку протокола OpenFlow v.1.0 – 1.5. Подробно с описанием протокола можно ознакомиться в источнике[3]. Кроме того, для увеличения скорости обработки сетевого трафика совместно с Open vSwitch можно использовать набор библиотек и драйверов Intel DPDK[4]. Для более глубокого изучения технологии SDN-сетей смотрите [1-5].

SANSET-узел обеспечивает управление виртуальными машинами. В задачи этого компонента входят, во-первых, обеспечение жизненного цикла (создание, удаление, управление) виртуальной машины, а, во-вторых, организация доступа виртуальной машины к коммуникационной сети. Кроме того, SANSET-узел предоставляет интерфейсы другим компонентам архитектуры для централизованного управления всей виртуальной инфраструктурой.

SANSET-узел имеет в своём составе SDN-коммутатор Open vSwitch с интегрированным набором библиотек и драйверов Intel DPDK; KVM – средство виртуализации уровня ядра Linux[5]; SANSET-Vi-Node – специальный набор библиотек, который обеспечивает централизованное управление и мониторинг виртуальных машин, запущенных на SANSET-узле.

SANSET-контроллер занимается централизованным управлением SANSET-узлами и SANSET-коммутаторами, а также оркестрацией[6] сетевых функций. SANSET-контроллер оперирует не виртуальными машинами, как это делает SANSET-узел, а сетевыми функциями, запущенными на SANSET-узлах. Этот компонент обеспечивает согласованную обработку сетевого трафика виртуальными машинами сетевой функции. При этом фактически SANSET-контроллер управляет SANSET-узлами и SANSET-коммутатором, добиваясь корректной работы сетевой функции.

SANSET-контроллер имеет в своём составе специализированный SDN-контроллер, набор скриптов SANSET-Vi, SANSET-Monitor и SANSET-Storage. Для обеспечения унификации ПО можно использовать SDN-контроллер RYU[7] с открытым исходным кодом. При повышенных требованиях к скорости обработки возможно заменить контроллер RYU на RunSDN [5].

SANSET-Vi – набор скриптов, который позволяет, используя SANSET-Vi-Node, осуществлять управление виртуальными машинами, а также передавать информацию об изменении вычислительной инфраструктуры программе SDN-контроллера, чтобы произвести корректировку необходимых сетевых настроек узлов и SANSET-коммутатора.

SANSET-Monitor – набор скриптов, который позволяет собирать статистическую информацию о запущенных виртуальных машинах на каждом SANSET-узле.

SANSET-Storage – специализированный набор скриптов для работы с образами виртуальных машин и с описанием сетевых функций. Предполагается, что образы хранятся в некотором файловом хранилище произвольной природы с возможностью удалённого доступа к нему. Профили сетевых функций могут быть помещены как в файловое хранилище, так и в произвольную базу данных, к который имеется удалённый доступ.

SANSET-оператор обеспечивает взаимодействие пользователя с системой в части управления сетевыми функциями и осуществляет визуализацию мониторинга системы.

SANSET-оператор состоит из набора скриптов SANSET-Operator для управления сетевыми функциями и набора скриптов SANSET-Visio для отображения текущего состояния сетевой функции.

Все логические элементы системы, кроме SANSET-оператора, могут быть совмещены на одном физическом сервере.

Специальное ПО архитектуры SANSET выполняет следующие базовые функции:

1. обеспечение жизненного цикла (создание, удаление, модификация и т.д.) профиля сетевой функции. Профиль сетевой функции содержит: описание вычислительных ресурсов (объём оперативной памяти, количество процессоров, тип процессоров и т.п.), которые выделяются виртуальным машинам сетевой функции при их старте; данные первичной инициализации (например, первичные ключи шифрования, настройки сетевых интерфейсов); IP-адреса и порты, к которым привязывается сетевая функция, а также другую информацию, необходимую для запуска виртуальных машин сетевой функции;
2. обеспечение жизненного цикла (создание, удаление, модификация и т.д.) образов виртуальных машин сетевой функции;
3. мониторинг сетевой функции;
4. принятие решений о масштабировании сетевой функции (увеличение и уменьшение количества виртуальных машин сетевой функции);
5. организация распределённой обработки сетевого трафика внутри сетевой функции.

Опишем подробно механизмы мониторинга, масштабирования сетевой функции, а также организации распределённой обработки сетевого трафика.

[2] OpenVSwitch documentation // Open vSwitch. [Электронный ресурс]. URL: https://docs.openvswitch.org/en/latest/ (дата обращения: 21.12.2018).

[3] OpenFlow Switch Specification. Version 1.5.1 (Protocol version 0x06). ONF TS-025. March 26, 2015. Pp. 283. [Электронный ресурс]. URL: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (дата обращения: 21.12.2018).

[4] Intel DPDK documentation. [Электронный ресурс]. URL: https://core.dpdk.org/doc/ (дата обращения: 21.12.2018).

[5] KVM documentation. [Электронный ресурс]. URL: https://www.linux-kvm.org/page/Documents (дата обращения: 21.12.2018).

[6] ETSI GS NFV-MAN 001. Network Functions Virtualisation (NFV); Management and Orchestration. V1.1.1. ETSI, Tech. Rep., December 2014. [Электронный ресурс]. URL: https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf (дата обращения: 21.12.2018).

[7] SDN-controller RYU [Электронный ресурс]. URL: https://ryu.readthedocs.io/en/latest/index.html (дата обращения: 21.12.2018).

Принципы распределённой обработки сетевого трафика

Общее представление устройств, построенных по архитектуре SANSET представлена на рисунке 1:

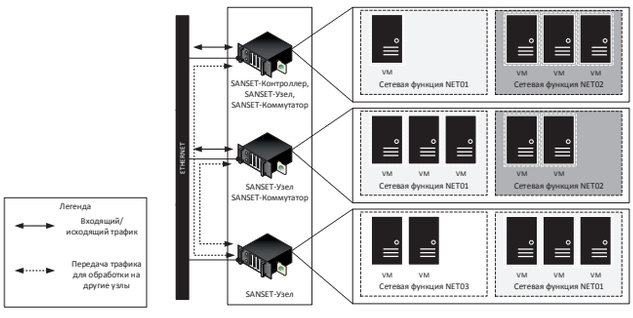

Рис. 1. Сетевые функции безопасности, построенные по архитектуре SANSET

Fig. 1. Network security functions built on the SANSET architecture

Обработка сетевого трафика в этом случае происходит следующим образом:

1. сетевой трафик поступает на SANSET-коммутаторы;
2. SANSET-коммутаторы, возможно обращаясь к SANSET-контроллеру, производят классификацию трафика и определяют правила для его обработки. Правила обработки транслируются в специальные OpenFlow-правила (OF-правила), которые устанавливаются в OF-коммутаторы. Когда входящий поток попадает под какое-либо существующее в коммутаторе правило, обращения к SANSET-контроллеру не происходит;
3. После классификации трафик обрабатывается в соответствии с OF-правилами и в результате отправляется в сторону какой-либо виртуальный машины какого-либо SANSET-узла. При этом сначала трафик средствами SANSET-контроллера отправляется в сторону нужного узла, а уже на OF-коммутаторе узла происходит его пересылка в нужный порт, к которому подключена виртуальная машина, назначенная контроллером для обработки данного потока;
4. Если после обработки в виртуальной машине генерируется некоторый исходящий поток трафика, то его сетевые пакеты с узла направляются в сторону SANSET-коммутатора и тот, в свою очередь, маршрутизирует их во внешнюю сеть.

Для управления сетевым трафиком и для балансировки нагрузки на виртуальные машины сетевой функции используется SDN-контроллер, который входит в состав SANSET-контроллера.

Процесс мониторинга сетевых функций

Мониторинг сетевой функции в архитектуре SANSET совмещён с основным циклом работы SANSET-контроллера. В процессе мониторинга участвуют SANSET-контроллер и SANSET-узел. Мониторинг можно разбить на следующие этапы:

Этап сбора информации. SANSET-контроллер N_{max} раз обновляет состояния виртуальных машин, запущенных на SANSET-узлах. Для этого контроллер опрашивает SANSET-узлы и получает с каждого из них список запущенных виртуальных машин с указанием для каждой виртуальной машины процент текущей загрузки центрального процессора и процент использования оперативной памяти. При этом SANSET-узел информацию о степени загрузки виртуальной машины собирает с помощью библиотеки управления виртуальными машинами *libvirt*[8]. Каждое обращение к узлам приводит к пересчёту текущих значений загрузки процессора и использования оперативной памяти. Пересчёт происходит по следующей формуле:

$$U_{N+1} = \frac{N \cdot U_N + u_{N+1}}{N+1},$$

здесь N – количество уже совершённых обновлений, U_N – текущее значение загрузки, U_{N+1} – новое значение загрузки и u_{N+1} – полученное в текущий момент значение загрузки. Легко понять, что итоговое значение загрузки $U = U_{N_{max}}$ будет средним арифметическим всех полученных значений. По завершению цикла обновления для каждой виртуальной машины ID_{VM}, запущенной на SANSET-узлах, будет сформирована запись:

$$VMstat_{ID_{VM}} = \{\text{cpustat} : C, \text{memstat} : M\},$$

где C – средний процент загрузки центрального процессора виртуальной машины, M – средний процент использования оперативной памяти виртуальной машины. В результате выполнения этапа будет сформирован список записей $VMstat = \left[VMstat_{ID_{VM}} \right]$.

Этап классификации виртуальных машин. После завершения этапа сбора информации о виртуальных машинах контроллер проводит классификацию всех виртуальных машин. В итоге формируются три списка состояний виртуальных машин. Первый список $L_{NotActive}$ состоит из тех записей $Rec_{ID_{VM}} \in VMstat$, для которых виртуальная машина с идентификатором ID_{VM} не привязана ни к одной активной сетевой функции. Второй список L_{ForDel} содержит записи $Rec_{ID_{VM}}$, для которых виртуальная машина ID_{VM} привязана к активной сетевой функции, но помечена к удалению. Третий список L_{Normal} содержит записи $Rec_{ID_{VM}}$, для которых виртуальная машина ID_{VM} привязана к активной сетевой функции и никак при этом не помечена. Обозначим через L_{Active} список, состоящий из записей $Rec_{ID_{VM}}$, для которых ID_{VM} привязанна к какой либо активной сетевой функции. Ясно, что

$$L_{Active} = L_{Normal} \cup L_{ForDel}.$$

Виртуальные машины из списка $L_{NotActive}$ физически удаляются. Для этого SANSET-контроллер по идентификатору виртуальной машины определяет SANSET-узел, на котором запущена эта виртуальная машина, и отправляет на этот узел команду об удалении виртуальной машины.

Список L_{Active} передаётся на следующий этап обработки.

Этап применения политик масштабирования. После формирования списка состояний виртуальных машин L_{Active} начинается этап применения к системе политик масштабирования каждой сетевой функции. Для этого для каждой сетевой функции последовательно выбираются политики масштабирования и в них передаются те элементы списка L_{Active}, которые

[8] LIBVIRT documentation // libvirt [Электронный ресурс]. https://libvirt.org/docs.html URL: (дата обращения: 21.12.2018).

привязаны к рассматриваемой сетевой функции. По завершении этапа процесс повторяется снова.

В качестве политик можно выделить политики двух типов. Первый тип – политика изменения количества виртуальных машин сетевой функции. Назовём этот тип политики «политикой горизонтального масштабирования». Второй тип – политика изменения объёма используемой оперативной памяти конкретной виртуальной машины. Этот тип политики назовём «политикой вертикального масштабирования».

Горизонтальное масштабирование сетевой функции

Рассмотрим подробнее процесс горизонтального масштабирования сетевой функции. Политика горизонтального масштабирования получает на вход:

идентификатор сетевой функции;

$L_{Active}^{ID_{VNF}}$ – список состояний виртуальных машин, привязанных к сетевой функции с идентификатором ID_{VNF}.

Кроме того, в политике задаётся предикат $\text{is\_heat}(x): \mathbb{R} \to \{\text{true}, \text{false}\}$. Помимо предиката параметрами сетевой функции являются границы для количества виртуальных машин, которые могут входить в сетевую функцию. Пусть N_{min} – минимально возможное количество виртуальных машин, а N_{max} – максимально возможное. Предполагается, что $0 \le N_{min} \le N_{max}$.

Предикат is_heat, границы N_{min} и N_{max} для каждой сетевой функции могут быть разными, и именно они определяют вариативность политики.

В начале для каждой виртуальной машины ID_{VM}, информация о которой содержится в списке $L_{Active}^{ID_{VNF}}$, вычисляется метрика $\propto$. Напомним, что каждая запись в этом словаре имеет вид:

$$VMstat_{ID_{VM}} = \{\text{cpustat} : C, \text{memstat} : M\},$$

где C – средний процент загрузки центрального процессора виртуальной машины ID_{VM}, M – средний процент использования оперативной памяти виртуальной машины ID_{VM}, то

$$\mu(ID_{VM}) = \begin{cases} 10 - \dfrac{C}{10}, & \text{is\_heat}(C) = \text{false} \\ 0, & \text{is\_heat}(C) = \text{true} \end{cases}.$$

Из формулы для метрики следует, что она принимает значения от 0 до 10. Значение 0 она принимает только в том случае, если средний процент загрузки C центрального процессора виртуальной машины равен 100, либо если средний процент загрузки обращает предикат $\text{is\_heat}(x)$ в истину. В связи с этим, предикат is_heat можно интерпретировать как критерий того, при каком проценте загрузки центрального процессора считать виртуальную машину достаточно «перегретой» для невозможности обработки новых потоков данных.

Напомним, что список $L_{Active}^{ID_{VNF}}$ представляет собой объединение двух непересекающихся списков:

$$L_{Normal}^{ID_{VNF}} \cup L_{ForDel}^{ID_{VNF}}.$$

После вычисления метрик производится обработка метрик для виртуальных машин из списка $L_{ForDel}^{ID_{VNF}}$. Если для записи $Rec_{ID_{VM}} \in L_{ForDel}^{ID_{VNF}}$ метрика $\mu(ID_{VM})$ равна 0, то виртуальная машина ID_{VM} физически удаляется, для этого SANSET-контроллер посылает на SANSET-узел, на котором запущена виртуальная машина, команду на её удаление.

Далее обрабатывается список $L_{Normal}^{ID_{VNF}}$.

Пусть $L_{Normal}^{ID_{VNF}} = \left(Rec_{ID_1}, Rec_{ID_2}, \ldots, Rec_{ID_N}\right)$,

тогда рассмотрим числа

$$S = \sum_{i=1}^{N} \mu(ID_i),$$

$$\mu_{min} = \min_{1 \le i \le N} \mu(ID_i),$$

при этом пусть $i_{min} = \text{argmin}_{1 \le i \le N} \mu(ID_i)$. Заметим, что минимум $\propto_{min}$ может достигаться на нескольких записях. Тогда в качестве i_{min} берётся номер любой записи, на которой достигается этот минимум.

Если $N < N_{max}$ и $S = 0$, либо $N < N_{min}$, то принимается решение масштабировать сетевую функцию «вверх», т.е. увеличить число виртуальных машин в сетевой функции. Сначала SANSET-контроллер определяет, существуют ли у этой сетевой функции виртуальные машины, которые ещё не были удалены физически, но присутствуют в списке на удаление. Если такие машины существуют, то контроллер их просто исключает из этого списка, и они становятся активными. Иначе начинается процедура физического добавления виртуальной машины. Для этого SANSET-контроллер опрашивает все SANSET-узлы и получает с них количество свободных ресурсов и величину их загрузки, например, в процентах использования центрального процессора и оперативной памяти. На основе этих данных выбирается наименее загруженный узел, отвечающий требованиям профиля сетевой функции, и ему посылается команда на старт новой виртуальной машины для сетевой функции. В результате SANSET-узел возвращает на контроллер идентификатор запущенной виртуальной машины. Предусматривается, что идентификатор отправляется сразу после его сформирования узлом. При этом процесс старта виртуальной машины продолжается на узле в асинхронном режиме. Только что запущенная виртуальная машина помечается тэгом $waited$, на основе которого SANSET-контроллер строит классификацию виртуальных машин.

Если $N > N_{min}$ и $S - \mu_{min} \ge (N-1) \cdot 5$, либо $N > N_{max}$, то принимается решение о масштабировании виртуальной функции «вниз», т.е. можно удалить наименее загруженную виртуальную машину сетевой функции. Поясним идею первого условия. Пусть система изменит количество виртуальных машин сетевой функции. Как это сделать наиболее оптимально? Предлагается выбрать простую стратегию – удалить наименее загруженную виртуальную машину. Если это будет сделано, то трафик, который обрабатывается этой виртуальной машиной, должен распределиться на оставшиеся узлы. При этом если оставшиеся машины сильно загружены, это приведёт к их перегрузке. Чтобы этого не произошло, выбирается следующая стратегия: совокупная загрузка оставшихся узлов должна быть ниже средней, а, значит, сумма их метрик должна быть выше средней.

Описанная стратегия конкурирует со стратегией, при которой уменьшение количества виртуальных машин происходит, когда загрузка узлов становится выше средней. Однако в этом случае важна не только средняя загрузка, но и степень отличия загрузки каждого узла от среднего значения, т.е. дисперсия загрузки. Например, если список метрик равен $(10, 3, 3)$, то

средняя загрузка машин будет равна $\frac{16}{3} > 5$. Тогда эта стратегия предлагает удалить первую виртуальную машину с метрикой 10. Но в этом случае даже незначительный рост трафика приведёт к ещё большей загрузки оставшихся виртуальных машин, а значит в итоге потребуется запускать новую виртуальную машину. В итоге возникнет эффект «мигания»: когда виртуальные машины будут быстро стартовать, а потом тут же уничтожаться системой. Выбранная же стратегия позволяет нивелировать этот эффект. В описанном примере средняя метрика оставшихся виртуальных машин слишком мала по сравнению со средней метрикой для трёх узлов. Как только метрика оставшихся машин поднимется до средней, то запаса их ресурсов хватит, чтобы распределить между собой новый поток трафика.

Для полноты описания масштабирования «вниз» нужно описать механизм удаления виртуальной машины. Предлагается её не удалять, а только включить в список на удаление. Такое «отложенное» удаление, во-первых, препятствует описанному эффекту «мигания» виртуальных машин, если вдруг внешняя обстановка с сетевыми потоками стремительно изменяется, а, во-вторых, она позволяет дождаться завершения активных сессий, и поэтому не происходит их обрыва.

Результатом применения к сетевой функции политики горизонтального масштабирования является список метрик:

$$CPU_{metrics} = \left[\left\{ ID_{VM} : \mu\left(ID_{VM}\right) \right\} 4; \text{для всех} Rec_{ID_{VM}} \in I_{Normal}^{ID_{VNF}} \right].$$

В дальнейшем этот список будет использован программой SDN-контроллера для балансировки нагрузки трафика на виртуальные машины сетевой функции.

Вертикальное масштабирование сетевой функции

Вертикальное масштабирование сетевой функции предполагает добавление оперативной памяти к тем виртуальным машинам сетевой функции, которые уже исчерпали выделенную им начальную квоту. Политика вертикального масштабирования получает на вход:

идентификатор сетевой функции;

$L_{Active}^{ID_{VNF}}$ – список состояний виртуальных машин, привязанных к сетевой функции с идентификатором ID_{VNF}.

Параметры политики вертикального масштабирования следующие: предикат $\text{is\_overflow}\left(x\right) : \mathbb{R} \to \left\{\text{true}, \text{false}\right\}$, смысл которого почти полностью совпадает со смыслом предиката is_heat с тем лишь исключением, что он применяется к загрузке оперативной памяти; M_{max} – максимальное количество оперативной памяти, которое может быть выделено виртуальной машине; M_{min} – минимальное количество оперативной памяти.

Напомним, что $L_{Active}^{ID_{VNF}} = L_{Normal}^{ID_{VNF}} \cup L_{ForDel}^{ID_{VNF}}$. В политике используется только $L_{Normal}^{ID_{VNF}}$. Для каждой виртуальной машины ID_{VM}, информация о которой содержится в списке $L_{Normal}^{ID_{VNF}}$, вычисляется метрика $\propto$:

$$\mu\left(ID_{VM}\right) = \begin{cases} 10 - \frac{M}{10}, & \text{is\_overflow}\left(M\right) = \text{false} \\ 0, & \text{is\_overflow}\left(M\right) = \text{true} \end{cases}.$$

Далее для каждой виртуальной машины, у которой $\mu\left(ID_{VM}\right) = 0$, выполняется увеличение количества оперативной памяти. Для этого SANSET-контроллер получает от SANSET-узла, на котором работает виртуальная машина ID_{VM}, количество M_{node} доступной оперативной памяти самого узла, а также количество $M_{ID_{VM}}$ доступной оперативной памяти виртуальной машины. SANSET-контроллер даёт команду SANSET-узлу на установку нового значения

$$\min\left(M_{max}, M_{node} - M_{reserv}, \frac{3}{2} M_{ID_{VM}} \right)$$

оперативной памяти виртуальной машины, где M_{reserv} – количество оперативной памяти, резервируемое для работы гипервизора, величина которого зависит от используемого в системе гипервизора.

Для каждой виртуальной машины, у которой $\mu\left(ID_{VM}\right) > 5$ проводится уменьшение оперативной памяти. В этом случае SANSET-контроллер получает только значение $M_{ID_{VM}}$ и далее отправляет команду SANSET-узлу установить новое значение оперативной памяти виртуальной машины:

$$\max\left(M_{min}, \frac{2}{3} M_{ID_{VM}} \right).$$

Результатом применения политики является список записей $MEM_{metrics} = \left[\left\{ ID_{VM} : \mu\left(ID_{VM}\right) \right\} \text{для всех} Rec \in L_{Normal}^{ID_{VNF}} \right]$.

Этот кортеж метрик может в дальнейшем использоваться другими узлами архитектуры.

Процесс балансировки нагрузки на узлы сетевой функции

Балансировка нагрузки предполагает разбиение входящего трафика на сетевые потоки. Сетевые потоки по возможности равномерно распределяются на виртуальные машины сетевой функции. Для каждой сетевой функции указывается правило разбиения трафика на потоки. Правило определяет, какие параметры сетевого пакета участвуют в формировании потока. В силу того, что SDN-контроллеры обычно оперируют данными только до уровня L4 модели ISO/OSI, то в формировании потока могут участвовать, например, тип Ethernet-протокола (Ethernet type), аппаратные адреса получателя и отправителя, для IP-трафика – номер IP-протокола, адрес получателя, адрес отправителя, для TCP/UDP-трафика – транспортные порты получателя и отправителя. Выбор конкретной виртуальной машины для обработки сетевого потока осуществляется на основе метрики виртуальной машины. Список метрик виртуальных машин вычисляется SANSET-контроллером на этапе применения политик балансировки.

За балансировку нагрузки на узлы сетевой функции отвечает внутренняя программа SDN-контроллера, работающая на SANSET-контроллере. Рассмотрим схематично алгоритм её работы.

Предположим, что в программе SDN-контроллера задан некоторый внутренний MAC-адрес, а также IP-адрес, которые будем называть соответственно MAC-адресом и IP-адресом контроллера.

Пусть на интерфейс SDN-коммутатора приходит первый пакет сетевого потока, для которого отсутствует правило его обра-

ботки. В этом случае коммутатор отсылает пакет на SDN-контроллер и начинается цикл работы программы контроллера:

1. После получения сетевого пакета из него извлекается информация об имени физического порта, в который был получен пакет на коммутаторе.
2. Если имя представляет собой идентификатор виртуальной машины, и трафик из этого порта ранее не поступал, то происходит регистрация аппаратного адреса виртуальной машины, и в базе данных программы контроллера делается пометка о соответствии идентификатора виртуальной машины и MAC-адреса её интерфейса. Кроме того, производится удаление пометки *waiting* с виртуальной машины, если ранее такая пометка имелась. Теперь на этапе классификации виртуальных машин SANSET-контроллером эта машина будет классифицирована как *normal* . В завершении обработки пакета на коммутатор, принявший пакет, отправляется правило, которое указывает, что весь трафик идущий с MAC-адреса виртуальной машины отправлять на внешний интерфейс. В итоге это предотвратит в дальнейшем появление на SDN-контроллере трафика (с целью анализа) с этой виртуальной машины.
3. В дальнейшем описании имя порта не является идентификатором виртуальной машины. Если MAC-адрес получателя пакета привязан к какой-либо виртуальной машине, то SDN-контроллер даёт указание коммутаторам отправлять такой трафик в сторону порта, за которым находится этот адрес.
4. Если MAC-адрес является внутренним адресом SDN-контроллера, тогда такой трафик подвергается балансировке нагрузки. Для этого из пакета извлекается IP-адрес и TCP/UDP-порт получателя. В профиле каждой сетевой функции указывается пул IP-адресов и пул TCP/UDP-портов, которые обрабатываются этой сетевой функцией. Происходит поиск сетевой функции, которая обрабатывает IP-адрес и порт пакета. Если такой функции нет, то пакет просто игнорируется. В противном случае на основе метрик, вычисленных политикой горизонтального масштабирования, выбирается виртуальная машина с максимальной метрикой, и находится MAC-адрес этой виртуальной машины. В завершении обработки контроллер указывает коммутатору установить правило, которое, во-первых, позволяет подменить MAC-адрес пакета на MAC-адрес виртуальной машины, а во-вторых, отправить этот пакет в порт коммутатора, за которым находится виртуальная машина[9].

SDN-контроллер на все ARP-запросы, присланные на его IP-адрес, отправляет ARP-ответ со своим MAC-адресом. Во всех остальных случаях сетевой трафик блокируется.

Отказоустойчивость и балансировка нагрузки на SANSET-контроллер

В предложенной архитектуре одним из узких мест является SANSET-контроллер, т.к. он выполняет существенный объём работы по управлению компонентами системы и по масштабированию сетевых функций. Для обеспечения отказоустойчивости и балансировки нагрузки на SANSET-контроллер предлагается использовать идеи работы [2].

Пусть в системе имеется несколько SANSET-контроллеров, которые выполняют функции, описанные в разделе 2. Для синхронизации контроллеров предлагается использовать технологию децентрализованных распределённых реестров данных (блокчейн). Её применение даёт существенное преимущество: во-первых, система становится децентрализованной и увеличивается совокупная стабильность архитектуры, во-вторых, уменьшается вероятность навязывания решений по управлению системой со стороны вероятного злоумышленника даже при условии, если он завладеет некоторым узлами.

Для обеспечения бесшовной интеграции блокчейна предлагается хранить список состояний виртуальных машин сетевых функций (см. пункт 4) в едином реестре. Согласование реестра может происходить на основе алгоритмов консенсуса типа «Proof of Work». Использование других типов протоколов консенсуса видится нецелесообразным, т.к. в данном случае основным ресурсом являются именно вычислительные мощности. Выбор конкретной реализации децентрализованного распределённого реестр должен осуществляться на этапе проектирования системы, построенной по архитектуре SANSET.

Интеграция сетевых функций в архитектуру SANSET

Интеграция сетевых функция в архитектуру SANSET в целом не требует специальных доработок образов виртуальных машин. Первое и самое главное требование – поддержка используемой платформы виртуализации. В силу того, что базовой компонентой сетевой функции являются виртуальные машины, первичный образ, из которого будет разворачиваться виртуальная машина, должен поддерживать запуск на платформе используемого гипервизора. Таким образом, основное требование интеграции сетевой функции в архитектуру накладывает именно используемый гипервизор, а не сама архитектура.

Для управления настройками и первичной инициализацией образов виртуальных машин в архитектуре SANSET может использоваться набор скриптов cloudinit. Платформа позволяет передавать в образ виртуальной машины конфигурационный диск первичной инициализации, который в образе может использоваться для выполнения специфичных настроек виртуальной машины сетевой функции. Для использования конфигурационного диска важно, чтобы в составе образа виртуальной машины имелся набор скриптов cloudinit, который и будет обращаться к этому диску.

Интеграция сетевых функций в архитектуру SANSET предполагает настройку программы для внутреннего SDN-контроллера для балансировки сетевого трафика на узлы сетевой функции. При этом настройка предполагает определение правила разбиения трафика на потоки. Программа балансировки распределяет потоки между виртуальными машинами сетевой функции. Следует отметить, что средства защиты информации часто требуют специфических правил разбиения трафика на потоки. Так, например, для устройств, выполняющих анализ отдельных IP-пакетов поток может состоять только из одного IP-пакета. К таким устройствам в первую очередь отно-

[9] IEEE 802.1AB-2005 - IEEE Standard for Local and metropolitan area networks -- Station and Media Access Control Connectivity Discovery [Электронный ресурс]. URL: https://standards.ieee.org/standard/802_1AB-2005.html (дата обращения: 21.12.2018).

сятся межсетевые экраны и однонаправленные шлюзы. Другие сетевые устройства анализируют целиком TCP/UDP-сессии. В этом случае поток будет определяться параметрами сессии. В первую очередь это относится к средствам шифрования трафика и устройствам обнаружения и предотвращения вторжений. Многие средства защиты работают на более высоких уровнях модели ISO/OSI. Так межсетевым экранам веб-приложений требуется анализировать целиком http-запросы и http-ответы. В этих случаях зачастую могут возникнуть сложности с определением потока, т.к. существующие SDN-коммутаторы оперируют параметрами до 4 уровня модели ISO/OSI включительно[10]. Тогда при интеграции в архитектуру SANSET возникнет необходимость такого разбиения сетевого трафика на потоки, чтобы, с одной стороны, обеспечить равномерную загрузку сетевых устройств, а с другой стороны, не фрагментировать http-трафик, сделав его непригодным для анализа в такого рода устройствах.

Потенциал сертификации в России

С точки зрения сертификационной лаборатории устройство, построенное в соответствии с SANSET-архитектурой, состоит из:

1. аппаратной платформы,
2. виртуальных машин,
3. программного гипервизора,
4. средства управления вычислительной инфраструктурой,
5. средства управления коммутацией сетевого трафика,
6. средства коммутации сетевого трафика.

При решении задачи масштабирования сетевой функции фактически решается задача запуска нового экземпляра виртуальной машины из существующего образа. При этом образ виртуальной машины может быть полностью зафиксирован, и при старте платформа может проверять контрольные суммы, чтобы исключить подмену образа виртуальной машины.

Изменение набора сетевых функций не приводит к повторной сертификации всей платформы, т.к. такое изменение требует только добавления сертифицированного образа виртуальной машины в систему и изменения ряда настроек.

В задачи первичной сертификации в первую очередь входит:

1. Сертификация образов виртуальных машин сетевых функций. При этом каждый образ представляет собой не отдельное программное обеспечение средства защиты, а статический программный комплекс с фиксированной программной средой. Сертификация такого образа может потребовать меньшего количества ресурсов. К тому же, при использовании средства защиты в реальных системах не потребуется проверка среды выполнения средства защиты, упростится оценка влияния среды выполнения на его работу.
2. Сертификация гипервизора. Использование гипервизоров с открытым исходным кодом в составе уже сертифицированных операционных систем, основанных на ядре Linux.
3. Сертификация средства управления вычислительной инфраструктурой и средства коммутации сетевого трафика. В некоторых системах сертификации этого может и не потребоваться.
4. Сертификация средств коммутации сетевого трафика. Предлагается использовать программные коммутаторы с открытым исходным кодом Open vSwitch, что поможет упростить сертификацию такого коммутатора. Стоит также отметить, что в некоторых системах сертификации этого не потребуется.

Таким образом при однократной сертификации платформы возможно обеспечить доверенное наращивание ресурсов сетевых функций без изменения компонентов системы.

Заключение

В работе предлагается подход к построению узлов защиты информации. Платформа узла строится с использованием принципов виртуализации сетевых функций. Коммутация сетевого трафика строится на базе технологии программно-конфигурируемых сетей. Архитектура позволяет выполнять запуск различных сетевых функций, решающих различные задачи защиты сетей, и обеспечивать масштабирование сетевых функций, а также балансировку нагрузки на экземпляры виртуальных машин сетевой функции.

Архитектура SANSET позволяет строить легко масштабируемые и расширяемые узлы защиты информации. К тому же такая архитектура может упростить производство и сертификацию по требованиям регуляторов в области защиты информации, как самих узлов защиты сети, так и компьютерных систем, построенных на их основе.

Список использованных источников

[1] *Kim H., Feamster N.* Improving network management with software defined networking // IEEE Communications Magazine. *2013. V*ol. 51, Issue 2. Pp. 114-119. DOI: 10.1109/MCOM.2013.6461195

[2] *Kreutz D., Ramos F. M. V., Verissimo P., Rothenberg C.E., Azodolmolky S., Uhlig S.* Software-Defined Networking: A Comprehensive Survey // Proceedings of the IEEE. 2015. Vol. 103, Issue 1. Pp. 14-76. DOI: 10.1109/JPROC.2014.2371999

[3] *Feamster N., Balakrishnan H.* Detecting BGP configuration faults with static analysis // Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. Vol. 2 (NSDI'05). Vol. 2. USENIX Association, Berkeley, CA, USA, 2005. Pp. 43-56.

[4] *Назаров М.А.* Определение, основные понятия и архитектуры программно-конфигурируемых сетей - SDN (Software Defined Networking) // Информатизация и связь. 2015. № 4. С. 82-87. URL: https://elibrary.ru/item.asp?id=24853434 (дата обращения: 21.12.2018).

[5] *Shalimov A., Zuikov D., Zimarina D., Pashkov V., Smeliansky R.* Advanced study of SDN/OpenFlow controllers // Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13). ACM, New York, NY, USA, 2013. Article 1, 6 p. DOI: 10.1145/2556610.2556621

[6] *Nuopponen A., Vaarala S., Virtanen T.* IPsec Clustering // Security and Protection in Information Processing Systems. SEC 2004. IFIP — The International Federation for Information Processing / Y. Deswarte, F. Cuppens, S. Jajodia, L. Wang

[10] OpenVSwitch documentation // Open vSwitch. [Электронный ресурс]. URL: https://docs.openvswitch.org/en/latest/ (дата обращения: 21.12.2018).

(eds). Springer, Boston, MA, 2004. Vol. 147. Pp. 367-379. DOI: 10.1007/1-4020-8143-X_24

[7] *Alvarenga I.D., Rebello G.A. F., Duarte O.C.M.B.* Securing configuration management and migration of virtual network functions using blockchain // NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium. Taipei, 2018. Pp. 1-9. DOI: 10.1109/NOMS.2018.8406249

Поступила 21.12.2018; принята к публикации 05.02.2019;
опубликована онлайн 19.04.2019.

Об авторах:

Гузев Олег Юрьевич, исследователь, Центр научных исследований и перспективных разработок, ОАО «ИнфоТеКС» (127287, Россия, г. Москва, Старый Петровско-Разумовский пр-д, д. 1/23, стр. 1), кандидат технических наук, ORCID: http://orcid.org/0000-0002-0737-6559, oleg.guzev@infotecs.ru

Чижов Иван Владимирович, доцент, кафедра информационной безопасности, факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (119991, Россия, г. Москва, Ленинские горы, д. 1); старший научный сотрудник, Институт проблем информатики, Федеральный исследовательский центр «Информатика и управление» Российской академии наук (119333, Россия, г. Москва, ул. Вавилова, д. 40), кандидат физико-математических наук, ORCID: http://orcid.org/0000-0001-9126-6442, ichizhov@cs.msu.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Kim H., Feamster N. Improving network management with software defined networking. *IEEE Communications Magazine. 2013;* 51(2):114-119. (In Eng.) DOI: 10.1109/MCOM.2013.6461195

[2] Kreutz D., Ramos F. M. V., Verissimo P., Rothenberg C.E., Azodolmolky S., Uhlig S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE. 2015;* 103(1):14-76. (In Eng.) DOI: 10.1109/JPROC.2014.2371999

[3] Feamster N., Balakrishnan H. Detecting BGP configuration faults with static analysis. Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation. Vol. 2 (NSDI'05). Vol. 2. USENIX Association, Berkeley, CA, USA, 2005, pp. 43-56. (In Eng.)

[4] Nazarov M.A. Definitions, concept sand architecture of Software Defined Networking – SDN. *Informatization and communication*. 2015; 4:82-87. Available at: https://elibrary.ru/item.asp?id=24853434 (accessed 21.12.2018). (In Russ.)

[5] Shalimov A., Zuikov D., Zimarina D., Pashkov V., Smeliansky R. Advanced study of SDN/OpenFlow controllers. Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia (CEE-SECR '13). ACM, New York, NY, USA, 2013. Article 1, 6 p. (In Eng.) DOI: 10.1145/2556610.2556621

[6] Nuopponen A., Vaarala S., Virtanen T. IPsec Clustering. In: Deswarte Y., Cuppens F., Jajodia S., Wang L. (eds) Security and Protection in Information Processing Systems. SEC 2004. IFIP – The International Federation for Information Processing, vol. 147. Springer, Boston, MA, 2004; 147:367-379. (In Eng.) DOI: 10.1007/1-4020-8143-X_24

[7] Alvarenga I.D., Rebello G.A. F., Duarte O.C.M.B. Securing configuration management and migration of virtual network functions using blockchain. *NOMS 2018 – 2018 IEEE/IFIP Network Operations and Management Symposium.* Taipei, 2018, pp. 1-9. (In Eng.) DOI: 10.1109/NOMS.2018.8406249

Submitted 21.12.2018; revised 05.02.2019;
published online 19.04.2019.

About the authors:

Oleg Yu. Guzev, Researcher, Research and Development Center, JSC "InfoTeCS" (1/23 Build. 1 Staryj Petrovsko-Razumovskij proezd, Moscow 127287, Russia), Ph.D. (Engineering), ORCID: http://orcid.org/0000-0002-0737-6559, oleg.guzev@infotecs.ru

Ivan V. Chizhov, Associate Professor, Department of Information Security, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1, Leninskie gory, Moscow 119991, Russia); Senior Scientist, Institute of Informatics Problems, Federal Research Center «Computer Science and Control» of Russian Academy of Sciences (40 Vavilova St., Moscow 119333, Russia), Ph.D. (Phys.-Math.), ORCID: http://orcid.org/0000-0001-9126-6442, ichizhov@cs.msu.ru

All authors have read and approved the final manuscript.