V. V. Tsehanovsky, V. D. Chertovskoy
*Saint Petersburg Electrotechnical University «LETI»*

PROGRAM MODEL STRUCTURE OF OPTIMAL PLANNING PROCESS IN MULTILEVEL SYSTEMS

*Program realization of mathematical large dimensions model for manufacturing function was considered. The model describes business-process «manufacturing» with help homogenous method. Choice of program means system using suggested requirements was performed.*

**Mathematical description, homogenous method, planning process, requirements to structure, program means system**

УДК 681.3.06 (075.8)

А. В. Смирнов, К. А. Борисенко, Е. С. Новикова, А. В. Шоров, И. В. Петухов
*Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В. И. Ульянова (Ленина)*

# Модуль обработки сетевых данных для обнаружения инфраструктурных атак в облачной вычислительной среде OpenStack

*Представлены результаты проектирования модуля сбора и подготовки исходных данных компонента защиты облачной инфраструктуры от DDoS-атак. Разработанный модуль способен обрабатывать как внешний, так и внутренний относительно облачной инфраструктуры сетевой трафик. Это позволяет компоненту защиты обнаруживать и реагировать на DDoS-атаки, источником которых может быть как внешний узел, так и узел, входящий в состав облачной инфраструктуры. Приводятся оценки производительности модуля, показывающие, что он способен обрабатывать сетевой трафик, превышающий по мощности современные DDoS-атаки.*

**Безопасность облачных вычислений, инфраструктурные атаки, архитектура компонента защиты облачной инфраструктуры, сбор и подготовка исходных данных, нагрузочное тестирование, система облачных сервисов OpenStack**

В настоящее время рынок облачных систем активно развивается; облачные технологии предлагают гибкие решения как для бизнеса, так и для частных пользователей, что обеспечивает постоянный рост спроса на них. При использовании облачных сервисов одним из важных моментов является их доступность пользователям. Согласно последним исследованиям, атаки типа «распределенный отказ в обслуживании» (Distributed Denial of Service – DDoS) входят в пятерку наиболее серьезных угроз информационной безопасности облачным сервисам [1]. Мощные DDoS-атаки влияют на работоспособность веб-сайтов государственных учреждений разных стран, ведущих IT-компаний, включая Amazon, Google, Yahoo, Microsoft и т. д. Большая часть современных систем защиты от DDoS-атак рассматривает облачную систему как единое целое, при этом не учитывается внутреннее устройство облака [2]. Такой подход обеспечивает универсальность системы защиты и ее независимость от реализации конкретной инфраструктуры облачных вычислений. Очевидным недостатком таких систем является невозможность детектирования инфраструктурных атак, источник и цель которых расположены внутри самого облака. Внутренние атаки, источником которых являются непосредственно «облачные» узлы, могут быть обнаружены установкой на каждую виртуальную машину специального программного обеспечения, выполняющегося в фоновом режиме и

собирающего данные о состоянии системы [3]. Эти данные отправляются в компонент безопасности, где анализируются, и на основе полученных результатов выявляются угрозы.

В статье описана архитектура компонента защиты системы облачных сервисов от DDoS-атак, которая позволяет реагировать на атаки, источники которых расположены как вне, так и внутри облачной инфраструктуры сетевого трафика; при этом не требуется установка каких-либо дополнительных сенсоров безопасности, запущенных в фоновом режиме в виртуальных машинах облачной инфраструктуры. Однако для реализации этой возможности необходимо учитывать особенности сетевой архитектуры системы облачных сервисов для определения расположения сенсоров, обрабатывающих как внутренний, так и внешний относительно «облака» трафик. В статье представлены результаты разработки модуля сбора и подготовки исходных данных для компонента защиты инфраструктуры облачных вычислений в системе OpenStack.

Статья структурирована следующим образом. Вначале описывается архитектура компонента защиты инфраструктуры облачных вычислений, сформулированы требования к модулю сбора и подготовки исходных данных. Далее анализируется архитектура системы облачных сервисов OpenStack и особенности маршрутизации сетевых потоков в ней. Затем обсуждаются особенности программной реализации модуля и проводятся экспериментальные оценки его производительности. В заключение подводятся итоги и определяются дальнейшие направления исследования.

**Архитектура компонента защиты инфраструктуры облачных сервисов от DDoS-атак.** Разрабатываемый модуль является частью компонента защиты облачных вычислительных сред от DDoS-атак. Отличительной чертой данного компонента является его способность обнаруживать и реагировать на атаки, источником которых может быть как вычислительный узел, расположенный внутри «облачной» инфраструктуры, так и узел, являющийся внешним относительно «облака». На рис. 1 представлена архитектура компонента защиты вычислительной инфраструктуры от DDoS-атак [4].

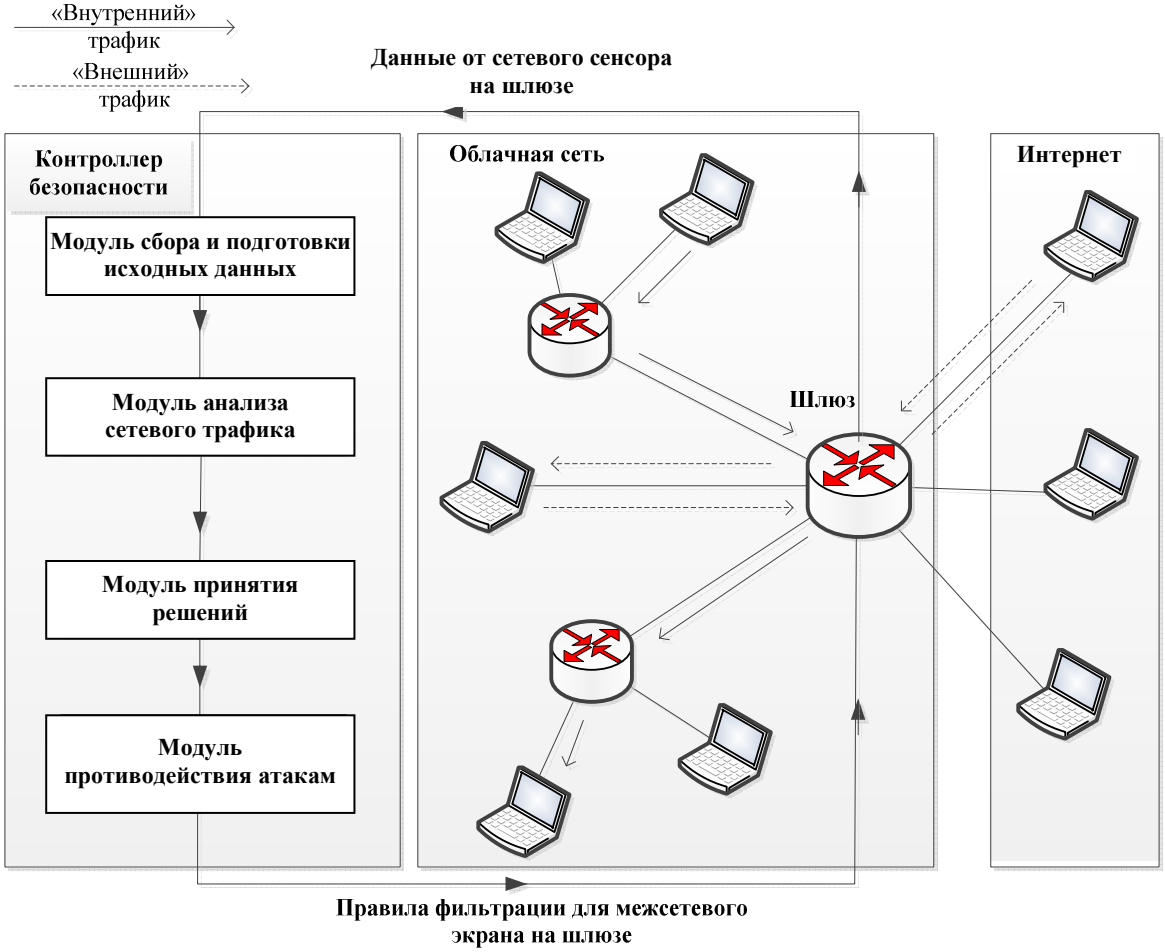


*Рис. 1*

Ключевыми модулями компонента защиты являются сенсор и контроллер безопасности, состоящий из модуля сбора и подготовки исходных данных, модуля анализа сетевого трафика, модуля принятия решений и модуля противодействия атакам. *Сенсор* располагается на шлюзе, через который проходит весь сетевой трафик – внутренний и внешний, и отвечает за сбор и отправку на контроллер безопасности данных о сетевых потоках. *Контроллер безопасности* – это модуль, который осуществляет сбор данных о сетевых потоках, их подготовку и анализ, по результатам которого принимает решение о легитимности сетевого трафика. В случае фиксации атаки отправляется команда на межсетевой экран для блокировки вредоносного трафика. *Модуль сбора и подготовки исходных данных* отвечает за прием и хранение данных о сетевых потоках. Кроме того он преобразует полученные данные в формат, необходимый для модуля принятия решений, в основе которых лежат методы интеллектуальной обработки данных. *Модуль принятия решений* выдает информацию о типе сетевого потока (легитимный или вредоносный) с указанием возможного типа атаки: атака с помощью переполнения пакетами SYN (SYN-флуд); атаки на основе протокола HTTP (HTTP-флуд); атаки на основе протокола ICMP (ping-флуд). На основе полученной информации *модуль противодействия* отправляет команду на межсетевой экран для противодействия конкретному типу атаки.

Очевидно, что для данного архитектурного решения компонента защиты важно правильное расположение сетевых сенсоров, осуществляющих сбор как внутреннего, так и внешнего относительно «облака» сетевого трафика. Расположение сетевых сенсоров зависит от сетевой архитектуры конкретной облачной инфраструктуры, что делает модуль сбора и подготовки исходных данных зависимым от конкретной реализации системы облачных сервисов, в то время как остальные модули системы универсальны.

Время реакции сенсора и контроллера безопасности должно быть минимальным, приближенным к реальному времени. Анализ современных DDoS-атак показал, что интервал времени от момента захвата трафика до момента противодействия должен быть меньше 10 с для оперативной реакции на атаку [4]. Эксперименты показали, что интервал, равный 4 с, достаточен для поддержания высокого уровня точности определения типа потоков и для быстрой ответной реакции на нежелательный трафик.

В настоящий момент модели, выполняющие классификацию сетевого трафика, используют следующие параметры: адрес источника; адрес назначения; порт источника для UDP и TCP; порт назначения для UDP и TCP; размер потока в байтах; количество потоков.

Таким образом, достаточно собирать информацию о сетевых потоках, не разбирая содержимое самих пакетов. Это позволяет повысить эффективность обработки сетевых данных за счет снижения объема передаваемых данных между модулями компонента защиты от DDoS-атак и уменьшения вычислительных ресурсов на их обработку.

**Система облачных сервисов OpenStack.** Система облачных сервисов OpenStack представляет собой систему взаимосвязанных проектов с открытым исходным кодом, что во многом объясняет его широкое распространение [5]. В программный состав OpenStack входят различные компоненты, отвечающие за создание и управление виртуальными машинами, контроль и распределение нагрузки между физическими машинами (проект Nova), создание сетевых подключений (проект Neutron), облачных хранилищ различного вида (проекты Swift и Cinder), идентификацию и аутентификацию пользователей (проект KeyStone) и т. д. Некоторые из этих компонентов обязательны для установки, остальные – устанавливаются по необходимости.

Физическая архитектура системы OpenStack в общем случае может быть представлена в виде трех компонентов: вычислительного узла (Compute node), представляющего собой основной сервер, на котором хранятся и выполняются виртуальные машины; коммутирующего узла (Network node) – сервера, отвечающего за весь сетевой трафик внутри облака и связь с Интернетом; контроллера (Controller), управляющего всем облаком. Сеть управления – это внутренняя виртуальная локальная сеть, в которой происходит служебное общение различных компонентов OpenStack. Туннельная сеть – это тоже виртуальная локальная сеть, но предназначена она для общения виртуальных машин между собой и для выхода в Интернет. При необходимости число вычислительных и других узлов может быть увеличено, причем все компоненты не обязательно должны находиться в пределах одной локальной сети, а могут быть физически разнесены по разным дата-центрам.

Способ физического размещения серверов определяет внутреннюю сетевую архитектуру системы OpenStack. В самой распространенной и

наиболее простой конфигурации, когда серверы находятся в одной локальной сети с фиксированными адресами, общий вид схемы маршрутизации сетевых потоков представлен на рис. 2 [5].

За соединение между узлами отвечает один из двух туннелей: OVS Tunnel Bridge или OVS VLAN Bridge. Ключевым элементом, отвечающим за обработку всего трафика, является модуль qrouter, который входит в состав коммутирующего узла Network. Самым сложным случаем для настройки и по сетевой архитектуре является вариант распределенного размещения компонентов. Данный вариант обладает большой гибкостью, но вместе с тем требует достаточно сложной сетевой настройки и поэтому на практике встречается редко. В этом случае архитектура коммутирующего узла Network остается неизменной, основные изменения касаются вычислительного узла Compute.

По аналогии с коммутирующим узлом в состав узла Compute добавляется модуль qrouter (рис. 3 [5]). В этом случае все сетевые потоки внутри удаленного вычислительного компонента будут обрабатываться локальным модулем qrouter.

Таким образом, после рассмотрения внутренней сетевой архитектуры OpenStack становится очевидным, что во всех вариантах построения облачной инфраструктуры присутствует сетевой модуль qrouter. Модуль qrouter представляет собой виртуальный программный роутер, разработанный на базе программного многоуровневого коммутатора с открытым кодом Open vSwitch (OVS) [6]. Учитывая, что независимо от архитектуры конкретного облака сетевой трафик обрабатывается одним из таких программных коммутаторов, сбор исходных данных для компонента защиты от инфраструктурных атак следует осуществлять с помощью модуля qrouter.

Дополнительным преимуществом данного решения является тот факт, что модуль qrouter имеет встроенный сетевой сенсор NetFlow [6], [7]. Данный сенсор выделяет из проходящего сетевого трафика потоки, которые характеризуются

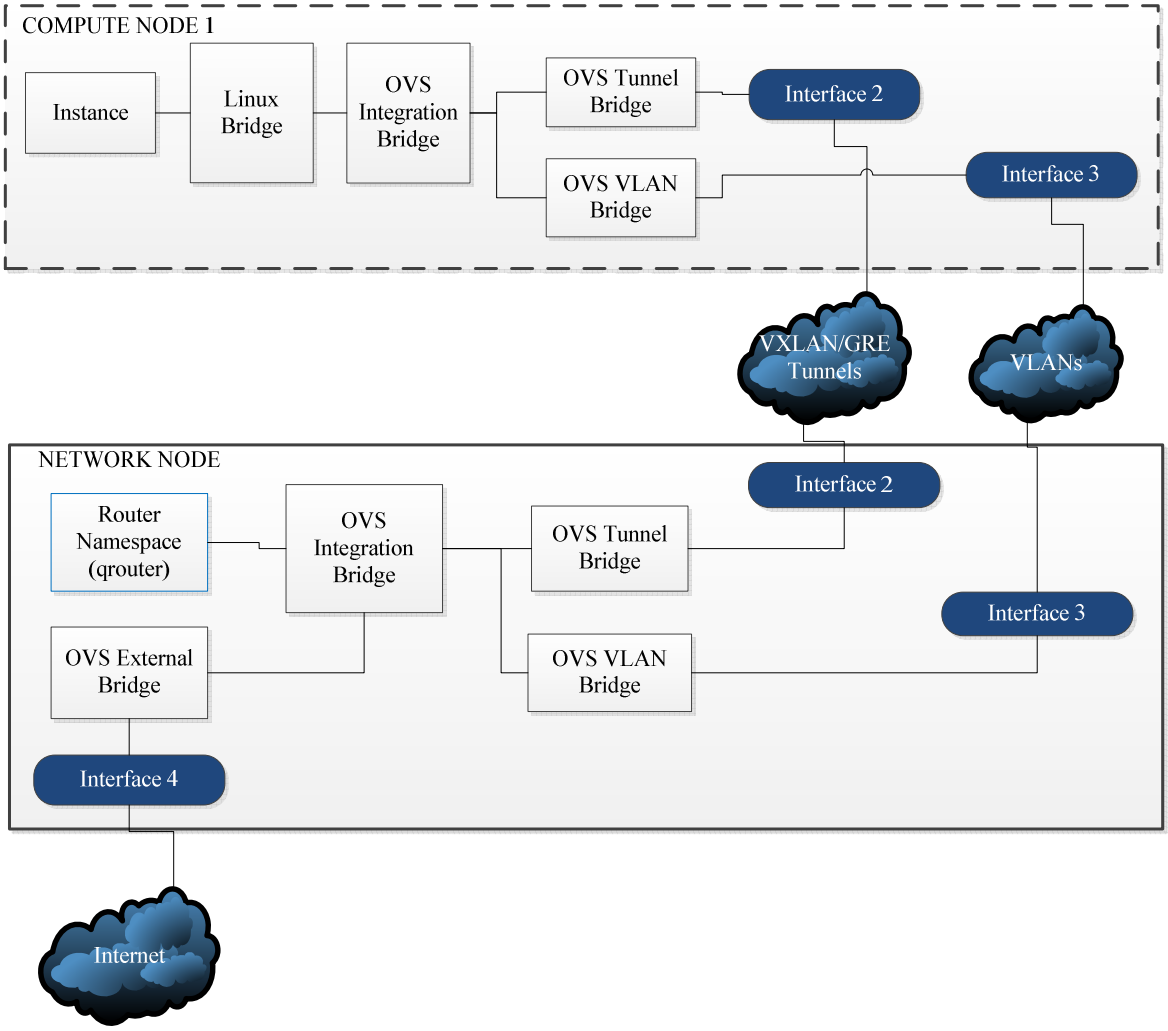


*Рис. 2*

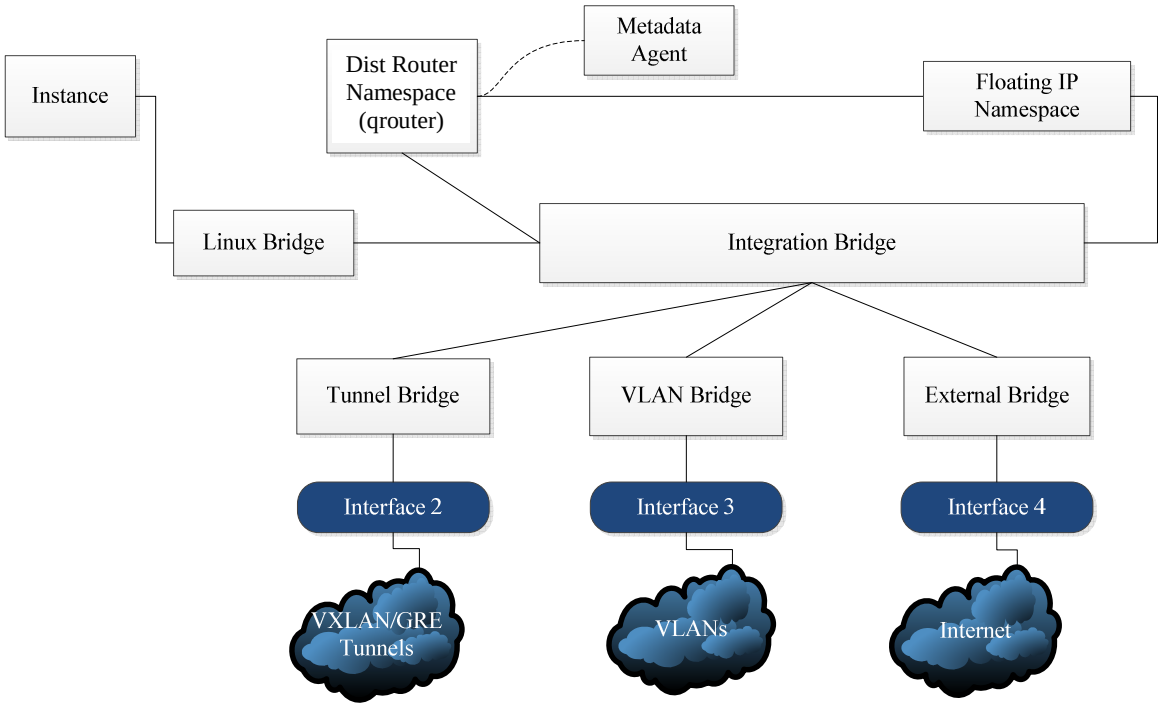


*Рис. 3*

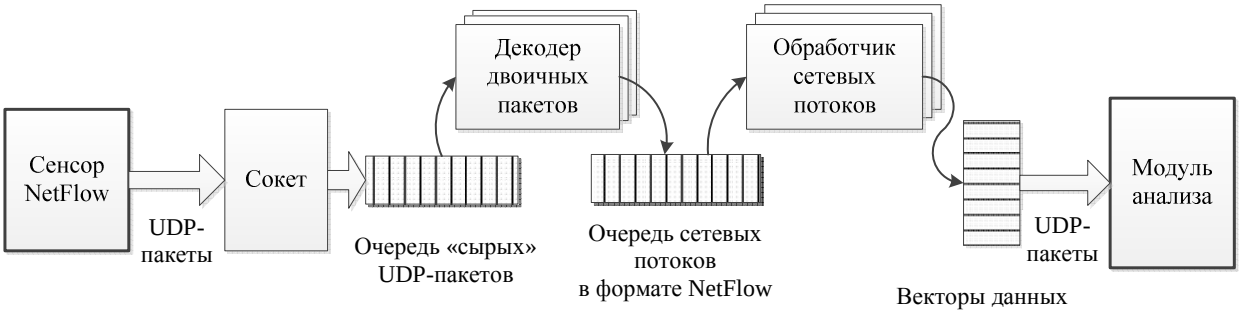


*Рис. 4*

следующими параметрами: адрес источника, адрес назначения, порт источника для UDP и TCP, порт назначения для UDP и TCP, тип и код сообщения для ICMP, номер протокола IP и т. д., т. е. предоставляет все необходимые на текущий момент данные для компонента защиты.

**Особенности программной реализации модуля сбора и подготовки исходных данных.** Для обеспечения своевременной реакции на потенциально опасный сетевой трафик модуль сбора исходных данных (коллектор) должен представлять собой многопоточное приложение, обеспечивающее распределение вычислительной нагрузки между всеми доступными ядрами процессора. Было принято решение каждый этап обработки данных – принятие сетевых пакетов, их обработку, формирование векторов данных для классификаторов – выполнять в независимом потоке, что позволит распределить вычислительные мощности между ними так, чтобы обработка происходила с минимальными задержками. Для выполнения этих требований предлагается следующая последовательность обработки исходных данных (рис. 4). На первом этапе UDP-пакеты, поступающие от сенсора NetFlow, сохраняют в памяти коллектора. Это нужно делать максимально быстро, иначе часть пакетов может быть утеряна. Пакеты представляют собой набор бинарных данных в формате протокола NetFlow v. 5. Полученные пакеты помещаются в очередь бинарных или «сырых» пакетов. Далее бинарные данные преобразуются в исходные данные об отдельных потоках. Этот разбор пакетов осуществляется в многопоточном режиме, полученные данные вновь помещаются в очередь NetFlow-потоков. Очередь потоков NetFlow также обрабатывается в многопоточном режиме, на выходе формируются векторы данных для классификаторов. Эти векторы имеют следующий формат: <количество байт, количество пакетов, число уникальных пар адресов и портов источника>.

Формируемые векторы данных передаются далее модулю принятия решений и записываются в лог-файлы.

Модуль сбора и подготовки исходных данных разработан с помощью программной платформы Node.js. Программная платформа Node.js предназначена для реализации высоконагруженных и легко масштабируемых веб-приложений реального времени. Использование неблокируемых сокетов и асинхронного событийно-ориентированного подхода позволяет получить высокие показатели быстродействия.

Для поддержи многоядерных процессоров была использована стандартная библиотека Node.js – Cluster. Она позволяет управлять работой скрипта на нескольких ядрах процессора. Работа с сетью реализована на низкоуровневых сокетах Datagram Sockets. Асинхронные очереди взяты из стандартной библиотеки Async.

Модуль обработки сетевых потоков может функционировать в двух режимах – отладочном и основном (рабочем). В отладочном режиме на экран выводится отладочная информация о принятых пакетах и сформированных векторах. В основном режиме модуль работает в фоновом режиме (режиме «демона»). По умолчанию он запускается в основном режиме, для запуска в режиме отладки необходимо в командной строке при запуске скрипта программы указать ключ debug.

**Оценка производительности разработанного модуля.** Время реакции компонента защиты облачной инфраструктуры от DDoS-атак зависит и от времени, затрачиваемого модулем сбора и подготовки исходных данных на обработку сетевых потоков. В работе оценивались следующие показатели производительности: загрузка ЦПУ, потребление сетевых ресурсов, временной интервал с момента получения данных от сенсора NetFlow до их преобразования во входные векторы модели классификации. Потребление оперативной памяти модулем зависит от настроек платформы Node.js, поэтому детально не рассматривалось. Производительность оценивалась с помощью методик нагрузочного тестирования системы, которое является одним из подтипов тестирования производительности системы.

Для проведения нагрузочного тестирования были реализованы сценарии различных инфраструктурных атак, которые в дальнейшем могут быть использованы для оценки эффективности модуля принятия решений и тестирования моделей классификации сетевых потоков. Атаки проводились с использованием различных специализированных программ, предназначенных как для тестирования производительности серверов, так и для имитации DDoS-атак, таких, как, hping3, siege, ab и др. Данные программы реализуют принципы DDoS-атак, используемых на практике, что обеспечивает высокую точность результатов тестирования. Процесс тестирования был автоматизирован с помощью инструмента Ansible, позволяющего удаленно управлять конфигурациями серверов.

Для организации тестовых DDoS-атак была создана тестовая виртуальная сеть в OpenStack. Она состояла из 100 атакующих виртуальных машин, включающих один узел, управляющий атакой, и 5 атакуемых виртуальных машин. Недостатком этого решения является ограничение мощности атаки пропускной способностью сети и вычислительной мощностью тестового стенда. В связи с этим полученные в эксперименте результаты были дополнены данными, сформированными моделированием работы сенсора NetFlow без генерации реального сетевого трафика.

Очевидно, что для проверки производительности коллектора не обязательно генерировать реальный сетевой трафик, поскольку модуль работает с сетевыми потоками. Было решено протестировать модуль сбора и обработки исходных данных с помощью скрипта, который генерирует пакеты по протоколу NetFlow v.5, моделируя тем самым работу сетевого сенсора. Это решение позволило смоделировать DDoS-атаки высокой мощности (до 500 Гбит/с) и оценить максимальную производительность модуля. Скрипт, моделирующий сенсор NetFlow, был разработан на языке Python. Скрипт формировал пакеты, идентичные тем, которые генерирует сенсор, и отправлял их тестируемому модулю.

Эксперименты показали, что максимальная производительность сенсора и модуля сбора при работе в паре ограничена двумя факторами: пропускной способностью канала между коллектором и сенсором и мощностью сервера, на котором находится коллектор.

Данные от сенсора приходят в виде UDP-пакетов, пакет содержит максимальную информацию о 30 потоках (ограничение стандарта NetFlow v.5). Если сенсор обработал 240 000 потоков в секунду, то он отправит 8000 UDP-пакетов по 1506 байт каждый, которые создадут нагрузку в 92 Мбит/с. Следовательно, если модуль и сенсор находятся на разных серверах, то при 100-мегабитном канале между ними макси-

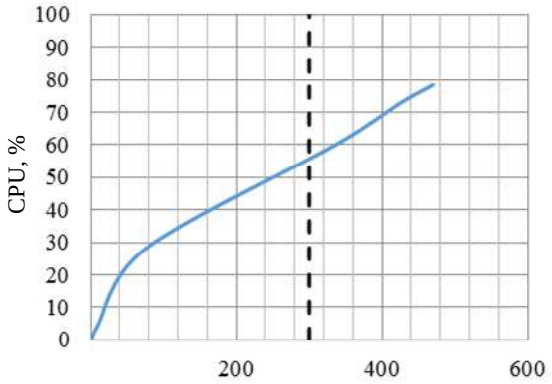


*Рис. 5*

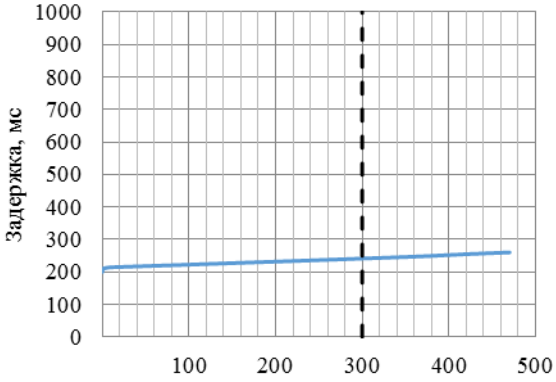


*Рис. 6*

мальное количество потоков равно примерно 240 тыс. При использовании гигабитного канала этот параметр возрастает до 2.4…2.5 млн потоков в секунду. В случае, когда модуль сбора данных и сенсор работают на одном сервере, т. е. пакеты не передаются по сети, ограничительным фактором является мощность сенсора. Модуль был протестирован на значениях до 4 млн входных потоков в секунду. При среднем размере потока 1 Мбайт значение мощности потока составит 3900 Гбайт/с. На рис. 5 и 6 показаны графики загрузки центрального процессора и временной задержки, вносимой модулем сбора и обработки исходных данных, в зависимости от мощности сетевого трафика. Штриховой линией на них отмечена максимальная зарегистрированная мощность DDoS-атак, которая на текущий момент равна 300 Гбит/с. С учетом того, что Node.js позволяет масштабировать данный скрипт на более мощный сервер, на текущий момент производительность скрипта можно считать достаточной.

В статье представлены результаты разработки модуля сбора и подготовки исходных данных для компонента защиты инфраструктуры облачных вычислений в системе OpenStack. На основе исследования сетевой архитектуры облачных сервисов OpenStack было предложено такое решение по сбору данных, которое не потребовало создания и установки дополнительных сенсоров сетевого трафика. Кроме того, это позволило избежать установки дополнительных сенсоров безопасности, функционирующих в виртуальных машинах пользователей облачной инфраструктуры. Последнее важно для компаний, политики безопасности которых строго регламентирует список используемого и запущенного программного обеспечения. В текущей реализации модуля в качестве сенсора сетевого трафика выступает сенсор NetFlow, который является фактически промышленным стандартом для учета сетевого трафика, что позволяет адаптировать модуль для использования в других облачных вычислительных средах. Оценки производительности модуля показали, что разработанный модуль способен обрабатывать сетевой трафик, превышающий по мощности современные DDoS-атаки.

Дальнейшие исследования будут связаны с проработкой модуля при распределенном расположении основных компонентов системы OpenStack.

Публикация выполнена в рамках государственной работы «Организация проведения научных исследований» базовой части государственного задания Минобрнауки России, а также проектной части государственного задания Минобрнауки России (ЗАДАНИЕ № 2.136.2014/К), поддержана грантом РФФИ №16-07-00625а и грантом Президента РФ. Также исследование выполнено при финансовой поддержке Фонда содействия развитию малых форм предприятий в научно-технической сфере.

## СПИСОК ЛИТЕРАТУРЫ

1. Top Threats: Cloud Security Alliance. URL: https://cloudsecurityalliance.org/research/top-threats, свободный.

2. Kaspersky DDoS Protection. URL: http://media.kaspersky.com/kaspersky-ddos-protection-data-sheet.pdf, свободный.

3. Lonea A. M., Popescu D. E., Tianfield H. Detecting DDoS Attacks in Cloud Computing Environment // Intern. J. of Computers Communications & Control. 2013. № 8 (1). P. 70–78.

4. Borisenko K., Kholod I., Shorov A. Modeling Framework for Developing and Testing Network Security Techniques against DDoS Attacks // Proc. of the 27th Intern. Conf. on Software Engineering and Knowledge Engineering. Pittsburg, USA: KSI Research inc., 2015. C. 715.

5. OpenStack Docs: Liberty: официальная документация OpenStack. URL: http://docs.openstack.org/liberty/, свободный.

6. OpenStack Docs: Scenario: Legacy with Open vSwitch. URL: http://docs.openstack.org/networking-guide/scenario_legacy_ovs.html, свободный.

7. NetFlow Export Datagram Format: описание структуры пакетов NetFlow. URL: ciscohttp://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html, свободный.

8. Choudhary S., Srinivasan B. Usage of Netflow in Security and Monitoring of Computer Networks // Intern. J. of Computer Applications. 2013. Vol. 68. P. 17–24.

A. V. Smirnov, K. A. Borisenko, E. S. Novikova, A. V. Shorov, I. V. Petukhov
*Saint Petersburg Electrotechnical University «LETI»*

THE NETWORK DATA PROCESSING MODULE FOR DETECTION OF THE INFRASTRUCTURE ATTACKS IN OPENSTACK CLOUD COMPUTING PLATFORM

*Currently the market of cloud services is developing fast. The paper presents a network data processing module for security component protecting cloud computing system again DDoS-attacks. The developed module processes both internal and external relative cloud infrastructure network traffic allowing thus security component to detect DDoS-attacks independently on the location of the attack source relative cloud infrastructure. The paper also presents the results of module load testing proving that the developed module is able to process volumes process network traffic exceeding the power of modern DDoS-attacks.*

**Cloud computing security, infrastructure attacks, architecture of cloud security component, input data collection and processing, load testing, OpenStack cloud computing system**

УДК 621.39, 629, 654

Я. А. Селиверстов, С. А. Селиверстов
*Институт проблем транспорта им. Н. С. Соломенко Российской академии наук*

# Метод построения пути субъективного предпочтительного следования

*Проводится анализ теоретических и прикладных решений, направленных на выявление критериев для выбора пути предпочтительного следования. Впервые формулируются понятия пути и области предпочтительного следования. Задачи прокладки пути предпочтительного следования и селективной фильтрации маршрутов разрешаются теоретико-множественной операцией пересечения и общим методом просеивания Сильва–Сильвестра соответственно. Приводятся практические примеры и даются рекомендации по дальнейшему использованию разработанных подходов.*

**Выбор пути; предпочтительный путь следования, область предпочтительного следования, интеллектуальные транспортные системы, умный город, городской транспортно-логистический мониторинг**

Непрерывное совершенствование современных городских информационно-аналитических систем с повсеместной интеллектуализацией городских логистических процессов, реализуемых на принципах Европейской концепции Смарт Сити, сегодня становится авангардом развития мегаполисов. Построение систем управления процессами городской мобильности населения, анализ и учет персонального транспортного поведения становятся возможными в границах направлений Smart Logistics и ITS[1]. Последние предполагают использование персональных навигационных логистических планировщиков, способных оптимизировать социально-экономическое и транспортно-логистическое поведение населения и прокладывать маршруты следования, с учетом субъективных предпочтений отдельного жителя. Способы прокладки подобных маршрутов потребуют разработки новых научных методов и подходов.

**Анализ предметной области.** Первой успешной попыткой предложить критерии выбора маршрута можно считать работу [1], в которой были сформулированы первый и второй принципы транспортного поведения, расширение которых до целевых ориентиров пользователя и транспортной системы было осуществлено в [2].

[1] ITS – Intelligent Transport Systems.