DOI 10.37882/2223–2966.2022.03–2.08

ПОСТРОЕНИЕ ВИРТУАЛЬНЫХ СЕТЕЙ С ИСПОЛЬЗОВАНИЕМ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

BUILDING VIRTUAL NETWORKS USING CRYPTOGRAPHIC ALGORITHMS

A. Kozlov
A. Mashihin

Summary. The article describes the principles of network configuration using the Docker tool and methods for developing a prototype of a system that allows you to create a virtual private network.

The developed system provides protection of data transmitted between the parties of VPN tunnels through the use of authenticated encryption with additional data.

This system implements the concept of traffic routing using cryptographic keys, providing the generation of key pairs using the elliptic curve Curve25519 on all network nodes necessary for the subsequent receipt of a shared secret key for data exchange using the Diffie-Hellman protocol on elliptic curves. The article presents the structure of the test stand and demonstrates the results of testing the system.

Keywords: virtual private network, Ansible, Fireguard, Jinja2, automation.

Козлов Александр Владимирович
К.т.н., доцент, МИРЭА Российский технологический университет
avkozlov@mirea.ru
Машихин Александр Юрьевич
Старший преподаватель, МИРЭА Российский технологический университет
mashihin@mirea.ru

Аннотация. В статье описаны принципы конфигурации сети с помощью инструмента Docker и методы разработки прототипа системы, позволяющей создавать виртуальную частную сеть.

Разработанная система обеспечивает защиту пересылаемых между сторонами туннелей VPN данных благодаря использованию аутентифицированного шифрования с дополнительными данными.

Данная система реализует концепцию маршрутизации трафика с использованием криптографических ключей, обеспечивая генерацию пар ключей с использованием эллиптической кривой Curve25519 на всех узлах сети, необходимых для последующего получения общего секретного ключа для обмена данными по протоколу Диффи-Хеллмана на эллиптических кривых. В статье приведена структура тестового стенда и продемонстрированы результаты тестирования системы.

Ключевые слова: виртуальная частная сеть, Ansible, Wireguard, Jinja2, автоматизация.

Введение

Интернет добился большого успеха за последние несколько десятилетий и предоставил совершенно новый способ доступа к информации и обмена ею, а также поспособствовал росту бизнеса. Его успех стимулировал огромный рост и широкое внедрение сетевых технологий и приложений. Однако вместе с тем Интернет стал причиной угроз конфиденциальности информации. Поэтому в настоящее время увеличивается потребность в инструментах защиты конфиденциальной информации в Интернете [1, 2, 3].

Виртуальная сеть является одним из таких инструментов, которые обеспечивают безопасность соединения за счет применения таких функций, как туннельное шифрование, аутентификация и др.

Целью данной статьи является разработка системы, позволяющей создавать защищенные виртуальные сети поверх общедоступных сетей.

Практическая значимость результатов работы заключается в том, что разработанная система позволит создавать защищенное соединение для безопасной передачи конфиденциальной информации через общедоступные сети для множества устройств и может быть использована на предприятиях или в личных целях.

Разработка и конфигурирование стендовой сети

В качестве стендовой сети, эмулирующей топологию сети организации, была выбрана сеть контейнеров Docker [4, 5]. Данные контейнеры образуют сеть, вклю-

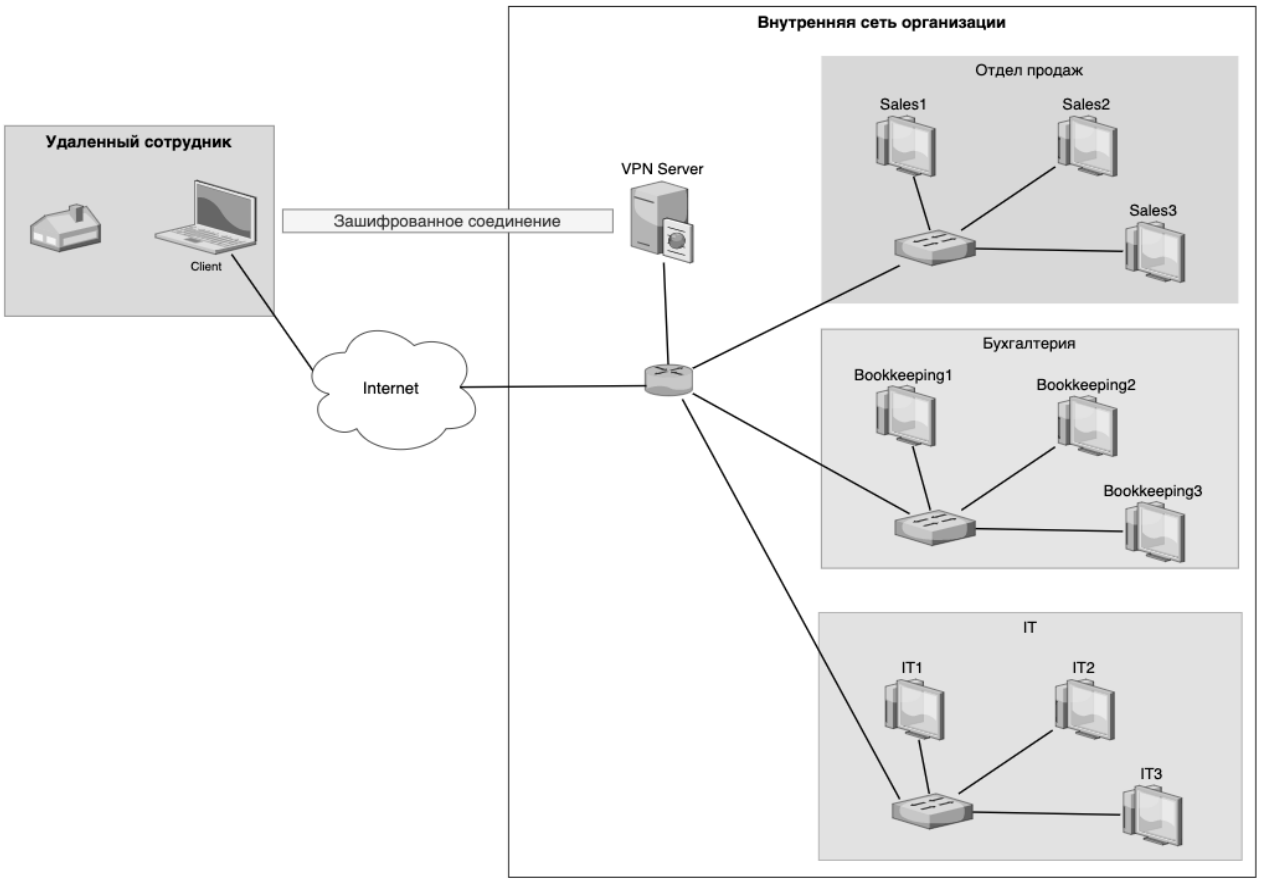

Рис. 1. Схема структуры организации

```
root@karina-VirtualBox:/home/karina/Desktop/vkr/Diplom# docker --version
Docker version 20.10.11, build dea9396
root@karina-VirtualBox:/home/karina/Desktop/vkr/Diplom# docker-compose --version
docker-compose version 1.29.2, build 5becea4c
root@karina-VirtualBox:/home/karina/Desktop/vkr/Diplom# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 20.04.3 LTS
Release:        20.04
Codename:       focal
```

Рис. 2. Вывод команд docker —version, docker-compose —version и lsb_release -a

чающую несколько подсетей, отведенных под различные отделы организации, например: отдел продаж, бухгалтерия, IT-отдел. Также в данной сети присутствует VPN-сервер, за счет которого организуется подключение удаленных устройств сотрудников к внутренней сети предприятия, а также доступ устройств внутренней сети к сети Интернет. Топология данной сети представлена на рисунке 1.

Таким образом, стендовая сеть состоит из контейнеров, эмулирующих конечные узлы, находящиеся в подсетях внутренней сети организации, контейнера, выполняющего роль VPN-сервера, и устройства, выступающего в роли VPN-клиента, осуществляющего подключение к внутренней сети организации из внешней сети.

Для того чтобы одновременно создать все необходимые контейнеры, а также сконфигурировать сети контейнеров, был выбран инструмент Docker Compose.

Для начала работы с Docker и Compose необходимо установить их на управляющем хосте. На рисунке

```
services:
  sales1:
    image: ubuntu_testbed2
    container_name: sales1
    networks:
      sales:
        ipv4_address: 172.18.0.2
      sales_vpn:
        ipv4_address: 172.23.0.2
    stdin_open: true
    tty: true
    cap_add:
      - NET_ADMIN
  sales2: ...
  sales3: ...

  book1:
    image: ubuntu_testbed2
    container_name: book1
    networks:
      bookkeeping:
        ipv4_address: 172.19.0.2
      bookkeeping_vpn:
        ipv4_address: 172.24.0.2
    stdin_open: true
    tty: true
    cap_add:
      - NET_ADMIN
  book2: ...
  book3: ...
```

```
vpn_server:
  container_name: vpn_server
  image: ubuntu_testbed2
  networks:
    vpn_server:
      ipv4_address: 172.21.0.5
    sales_vpn:
      ipv4_address: 172.23.0.5
    bookkeeping_vpn:
      ipv4_address: 172.24.0.5
    it_vpn:
      ipv4_address: 172.25.0.5
    remote_vpn:
      ipv4_address: 172.26.0.5
  stdin_open: true
  tty: true
  cap_add:
    - NET_ADMIN
    - NET_RAW

remote1:
  container_name: remote1
  image: ubuntu_testbed2
  networks:
    remote:
      ipv4_address: 172.22.0.2
    remote_vpn:
      ipv4_address: 172.26.0.2
  stdin_open: true
  tty: true
```

Рис. 3. Блок кода services файла docker-compose.yml

```
karina@karina-VirtualBox:~/Desktop/vkr$ ip -o addr show | awk '/inet/ {print $2, $3, $4}'
| grep -v inet6
lo inet 127.0.0.1/8
enp0s3 inet 10.0.2.15/24
docker0 inet 172.17.0.1/16
br-6423f01bc9c6 inet 172.20.0.1/24
br-f94e0a937b07 inet 172.22.0.1/24
br-0c55b8f267e9 inet 172.23.0.1/24
br-42758309225b inet 172.26.0.1/24
br-ff96a0b45f70 inet 172.21.0.1/24
br-f6924ca851d7 inet 172.19.0.1/24
br-73dcf0364151 inet 172.24.0.1/24
br-8b397134d2d8 inet 172.25.0.1/24
br-1cee9a80e091 inet 172.18.0.1/24
```

Рис. 4. Сетевые интерфейсы на основной машине

2 представлены версии установленных инструментов. Также на рисунке видно, что на управляющем хосте установлен дистрибутив Linux Ubuntu 20.04.

Так как контейнеры представляют собой конечные узлы стендовой сети с дистрибутивом Ubuntu 20.04, в качестве процесса будет выступать команда bash. Для создания данных контейнеров был использован официальный образ Ubuntu, размещенный в общедоступном репозитории Docker Hub.

Упомянутый выше образ не содержит заранее установленных базовых системных пакетов, таких как net-tools, iputils, а также Python, необходимых для работы

и тестирования стендовой сети, поэтому был создан файл Dockerfile.

Благодаря этому файлу с помощью команды docker build -t ubuntu_testbed2 был создан новый образ ubuntu_testbed2 из родительского образа ubuntu последней версии, в котором сначала был обновлен индексный файл пакетов, а затем установлены Python3, утилиты net-tools, iproute2, traceroute, iputils-ping и lsb-release.

Далее был создан файл docker-compose.yml. В данном файле в блоке networks были перечислены сети узлов с их адресами и шлюзами по умолчанию. В данном блоке были перечислены сети отделов продаж, бухгалтерии и IT-отдела организации. Кроме того, так как стендовая сеть состоит из контейнеров Docker, в блоке были указаны сети, необходимые для создания соединений между сервером и клиентами, которые в реальных условиях будут созданы в более сложных сетях, таких как внутренняя сеть организации и Интернет. Все сети являются сетями типа bridge для возможности обмена данными между контейнерами [6].

На рисунке 3 приведен фрагмент кода файла с описанием параметров сервисов.

Кроме используемого образа для каждого сервиса также были указаны IPv4-адреса, параметры stdin_open и tty для интерактивного режима и параметр NET_ADMIN для взаимодействия с сетевым стеком. Также для контейнера, выступающего в роли VPN-сервера, был добавлен параметр NET_RAW для возможности применять правила iptables.

Для запуска указанных в файле docker-compose контейнеров была выполнена команда docker-compose up.

При создании сети в Docker на основной машине создается соответствующий сетевой интерфейс. В каждой сети Docker основная машина играет роль шлюза по умолчанию. На рисунке 4 представлены IP-адреса, назначенные всем интерфейсам на основной машине.

Таким образом, стендовая сеть создана и сконфигурирована для тестирования системы.

Разработка прототипа системы

Для автоматизации процесса настройки и развертывания сервера и клиентов VPN была выбрана система управления конфигурациями Ansible.

Данная система управления конфигурациями осуществляет соединение с целевыми узлами при помощи драйвера подключения Ansible.

Ansible одновременно может работать с несколькими управляемыми узлами инфраструктуры, используя список узлов, также называемый «инвентарем» [7, 8, 11].

Команда ansible позволяет запускать специальные команды для групп устройств, однако можно запускать последовательно несколько команд, если указать их в файле формата YAML, который выполняется с помощью команды ansible-playbook.

Ansible также можно использовать вместе с механизмом шаблонов Jinja2. Ansible использует шаблоны Jinja2 для включения динамических выражений и доступа к переменным. Так, например, данный механизм полезен при создании шаблонов файлов конфигурации и развертывания их на разных устройствах с корректными данными, различными для каждого устройства, такими как IP-адреса, имена хостов и др.

Непосредственно для реализации VPN был выбран протокол WireGuard из-за его эффективности, а также удобства и простоты использования.

Протокол WireGuard обеспечивает безопасный сетевой туннель уровня 3 модели OSI между двумя конечными точками с использованием протокола пользовательских дейтаграмм (англ. User Datagram Protocol, сокр. UDP) в качестве транспортного протокола. Транспортные данные, такие как IP-пакеты, инкапсулированные в туннелях WireGuard, защищены с помощью аутентифицированного шифрования с дополнительными данными (англ. Authenticated Encryption with Additional Data, сокр. AEAD).

Протокол использует следующие эффективные криптографические примитивы [10, 11]:

- ChaCha20 — потоковый шифр с симметричным ключом;
- Poly1305 — код аутентификации сообщения, используемый для аутентификации соединений WireGuard;
- Curve25519 — эллиптическая кривая, используемая протоколом Диффи — Хеллмана на эллиптических кривых;
- SipHash — псевдослучайная хеш-функция на основе XOR, используемая для безопасного сопоставления ключей хеш-таблицы;
- BLAKE2 — функция криптографического хеширования, используемая для проверки данных.

Протокол WireGuard использует концепцию маршрутизации с использованием криптографических ключей (англ. Cryptokey routing). Это означает, что все конечные точки туннелей идентифицируются своим открытым ключом Curve25519.

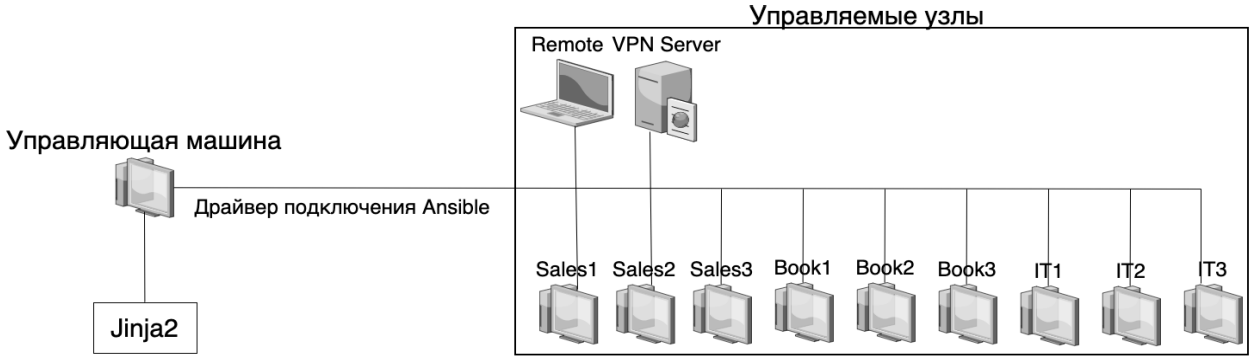

Рис. 5. Структурная схема системы построения виртуальных частных сетей

```
root@a2b3fb8f7163:/# ip -o addr show | awk '/inet/ {print $2, $3, $4}'
lo inet 127.0.0.1/8
eth1 inet 172.26.0.2/24
eth0 inet 172.22.0.2/24
```

Рис. 6. Вывод команды ip addr show на узле remote1

Протокол WireGuard четко разделен на две отдельные фазы:

- фаза обмена ключами, когда узлы обмениваются эфемерными значениями Диффи-Хеллмана на эллиптической кривой, вычисленных с использованием статических ключей, и вычисляют ключи AEAD;
- фаза передачи данных, на котором пользователи могут отправлять аутентифицированные и конфиденциальные транспортные данные с использованием ранее вычисленных ключей AEAD.

Для того, чтобы настроить туннель VPN, необходимо:

1. На каждом узле загрузить пакет WireGuard с помощью команды sudo apt install wireguard;
2. На каждом узле сгенерировать статическую пару 32-байтовых ключей с использованием эллиптической кривой Curve25519 и обменяться с другим участником соединения публичными ключами;
3. Каждому узлу обменяться адресной информацией с другим узлом;
4. На каждом узле создать файлы конфигурации, содержащие таблицы маршрутизации и при необходимости команды PostUp и PostDown, которые будут выполнены при включении и выключении интерфейса;
5. На всех узлах выполнить команду wg-quick up, которая загрузит файл конфигурации и активирует интерфейс WireGuard.

После выполнения перечисленных шагов безопасное соединение между узлами будет установлено.

Таким образом, система построения виртуальных сетей с использованием криптографических алгоритмов состоит из управляющей машины с установленным инструментом Ansible, который использует механизм Jinja2 для управления файлами конфигурации, подключенной к группе целевых узлов сети, на которых производится настройка и запуск протокола WireGuard. Структурная схема системы приведена на рисунке 5.

Для настройки и запуска протокола WireGuard на узлах сети сначала нужно установить Ansible на управляющей машине, а затем создать сценарии настройки протокола на сервере и клиентах VPN.

Тестирование работы системы

До настройки протокола на узлах было произведено подключение к контейнеру remote1 с управляющей машины. На рисунке 6 показаны IP-адреса на сетевых интерфейсах данного узла, полученные с помощью команды ip addr show. Как видно на рисунке, данный узел состоит в сетях 172.22.0.0/24 и 172.26.0.0/24.

После этого был совершен захват пакетов на интерфейсе br-42758309225b основной машины, соответствующий сети 172.26.0.0/24, с помощью Wireshark во время эхо-запроса от контейнера remote1 к узлу с адресом

```
root@a2b3fb8f7163:/# ping -c 1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=18.4 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 18.365/18.365/18.365/0.000 ms
```

Рис. 7. Вывод выполнения команды ping, осуществляющей посылку эхо-запроса с узла remote1 на узел с адресом 8.8.8.8 из внешней сети.

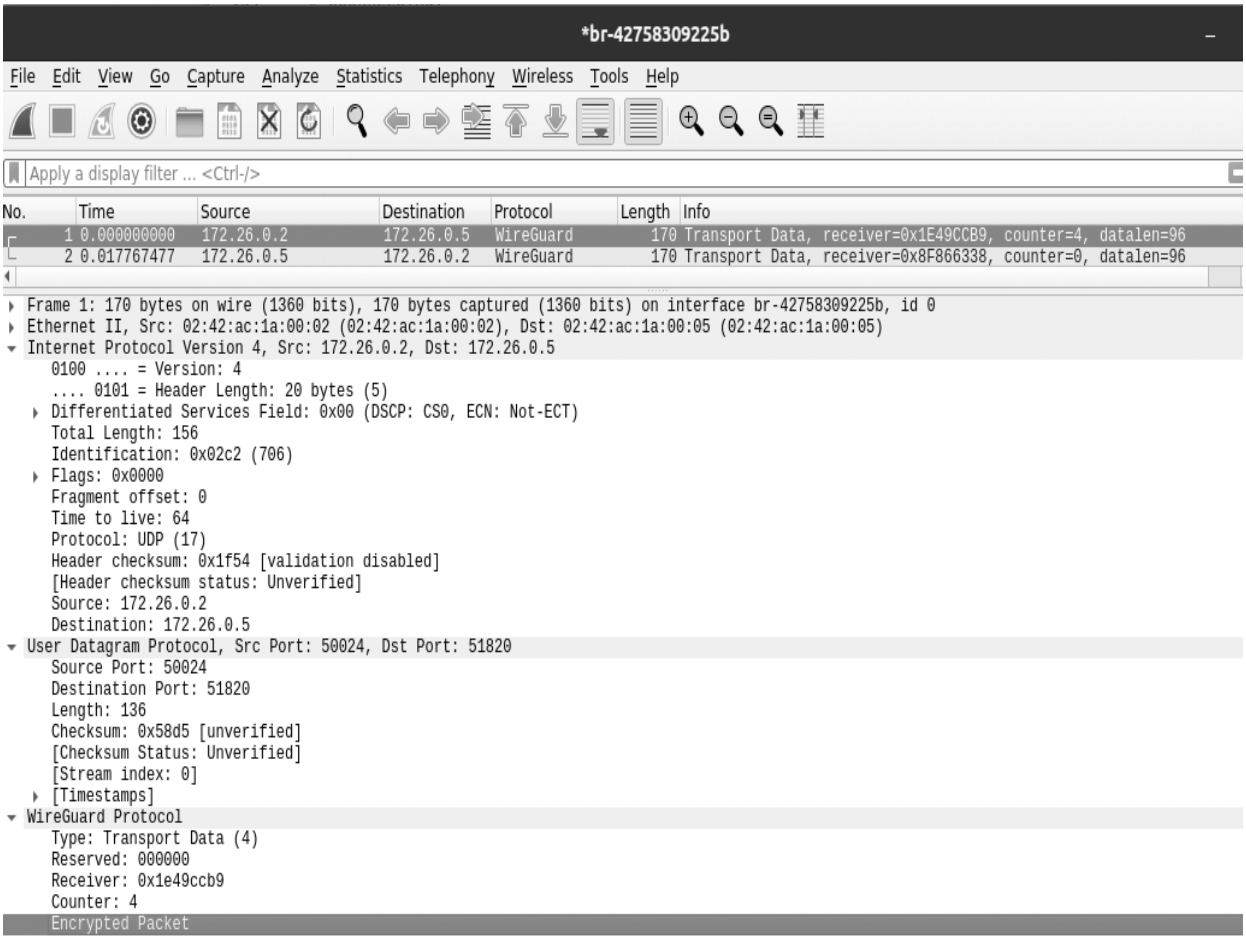

Рис. 8. Файл захвата пакетов на интерфейсе br-42758309225b основной машины

8.8.8.8. Как было указано выше, данная сеть обеспечивает соединение узлов remote1 и vpn_server. Вывод команды ping и файл захвата приведены на рисунках 7 и 8 соответственно.

Из последнего рисунка следует, что между клиентом и сервером виртуальной частной сети создан зашифрованный туннель. Анализатор трафика Wireshark определил, что пакеты между узлами передаются по протоколу Wireshark, однако сам пакет зашифрован, и нельзя определить, что два представленных на рисунке захваченных пакета являются запросом и ответом ICMP между клиентом VPN и узлом из внешней сети.

Таким образом, можно судить о том, что туннель VPN исправно функционирует.

- настройку виртуальной частной сети на узлах сервера и клиентов;
- защищенное соединение между сервером и клиентами.

ЛИТЕРАТУРА

1. Mohammad Taha Khan, Joe DeBlasio, Geoffrey M. Voelker, Alex C. Snoeren, Chris Kanich, and Narseo Vallina-Rodriguez. An Empirical Analysis of the Commercial VPN Ecosystem // 2018 Internet Measurement Conference (IMC '18), 2018–14 с.
2. Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. An Analysis of the Privacy and Security Risks of Android VPN Permission-enabled Apps // ACM Int. Measurement Conference (IMC), 2016–16 с.
3. В Сеть утекла база данных 21 миллиона пользователей VPN-сервисов — 2018 [электронный ресурс].— URL: https://ria.ru/20210301/utechka-1599382868.html (дата обращения 15.09.2021).
4. Streisand — 2021 [электронный ресурс].— URL: https://github.com/StreisandEffect/streisand (дата обращения 20.09.2021).
5. Algo VPN — 2021 [электронный ресурс].— URL: https://github.com/trailofbits/algo (дата обращения 20.09.2021).
6. Официальный сайт Virtual Box — 2021 [электронный ресурс].— URL: https://www.virtualbox.org/ (дата обращения 15.10.2021).
7. Официальный сайт VMware — 2021 [электронный ресурс].— URL: https://www.vmware.com/ (дата обращения 15.10.2021).
8. Junzo Watada Arunava Roy, Ruturadj Kadikar, Hoang Pham, Bing Xu. Emerging Trends, Techniques and Open Issues of Containerization: A Review // IEEE Access, 2019.— 30 с.
9. Mike Schwartz, Andy Dennis, Richard Bullington-McGuire. Docker for Developers. Develop and run your application with Docker containers using DevOps tools for continuous delivery // Packt Publishing, 2020.— 468 с.
10. Официальный сайт Cisco — 2021 [электронный ресурс].— URL: https://www.cisco.com/ (дата обращения 15.10.2021).
11. Официальный сайт Ansible — 2021 [электронный ресурс].— URL: https://docs.ansible.com/ (дата обращения 15.10.2021).

Журнал «Современная наука: актуальные проблемы теории и практики»

МИРЭА — Российский технологический университет