ПАТТЕРН ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ИНФОРМАЦИОННОЙ ИНФРАСТРУКТУРЫ ПРИ МИГРАЦИИ ОБРАЗОВ ВИРТУАЛЬНЫХ МАШИН

Корнеев Н. В.[1], Дикий А. Б.[2]

DOI: 10.21681/2311-3456-2025-2-29-40

Цель статьи: *разработка шаблонного механизма защиты для обеспечения безопасности информационной инфраструктуры при миграции образов виртуальных машин.*

Метод исследования: *анализ принципов виртуализации аппаратного обеспечения и процесса миграции образа виртуальной машины. Синтез сценария атаки отказ в обслуживании (DoS) в процессе миграции образа виртуальной машины в несовместимую среду. С использованием методов низкоуровневого программирования предложен новый механизм защиты путем вызова, анализа системных функций через высокоуровневые системные библиотеки и формирования специального слоя системных функций для принятия окончательного решения о миграции образа виртуальной машины. Специальный слой системных функций реализуется как сервис-решение в защищенной среде с использованием ssh протокола. Исследование выполнено путем натурного моделирования информационной инфраструктуры на основе Docker в средах с поддержкой контейнеризации, её развёртывания и тестирования при реализации сценария атаки.*

Результат: *проведен анализ угрозы возникновения технических сложностей при миграции образов виртуальных машин между поставщиками облачных услуг, обусловленных несовместимостью аппаратного и программного обеспечения и показана актуальность проблемы разработки универсальных шаблонных механизмов безопасности, называемых паттернами. Построена микросервисная архитектура для обеспечения безопасности информационной инфраструктуры при миграции образов виртуальных машин. Рассмотрен сценарий атаки отказ в обслуживании (DoS) в процессе миграции образа виртуальной машины в несовместимую среду. Разработан паттерн безопасности информационной инфраструктуры при миграции образов виртуальных машин на основе микросервисов, интегрированных в Docker-контейнер. Разработаны 3 микросервиса: проверки программного обеспечения; проверки аппаратной части; проверки гипервизора. Разработан специальный слой системных функций для принятия окончательного решения о миграции образа виртуальной машины, включающий: набор функций проверки версии VirtualBox; функцию проверки ресурсов системы; набор функций проверки совместимости файловой системы и доступного места на диске; функцию проверки совместимости оборудования и ресурсов; функцию проведения всех проверок и журналирования результатов. Разработан программный код микросервисов, включая коды для системных функций и специальных классов: CheckSoftware, CheckHardware, CheckHypervisor, Logs, обеспечивающих механизм защиты. Разработан алгоритм создания паттерна безопасности в виде Docker-контейнера, написан Dockerfile, создан Docker-образ и Docker-контейнер. Проведено тестирование микросервисов, которое показало успешную работу механизма защиты. Развёрнута система мониторинга процесса миграции образа виртуальной машины в несовместимую среду на базе открытого программного обеспечения и созданного Docker-контейнера, которая может быть использована в системах SIEM, а метрики ошибок миграции могут быть настроены через соответствующие события специальных классов, что дает возможность гибко настраивать сам паттерн и применять его для различной информационной инфраструктуры.*

Практическая ценность: *практическая значимость предлагаемого решения включает шаблонный механизм защиты в виде паттерна, который можно применить для обеспечения безопасности различной информационной инфраструктуры, состоящей из разного набора ВМ, гипервизоров и серверов, в том числе перенести разрабатываемое решение на любую отрасль: топливно-энергетическую, экономическую и не только, ввиду кроссплатформенности самого решения.*

Ключевые слова: *облачные вычисления, виртуализация, шаблон, атака отказ в обслуживании, несовместимая среда, специальный слой системных функций, ssh протокол, гипервизор, сервер, контейнер, журнал событий, система мониторинга.*

Введение

В современном информационном обществе виртуализация играет ключевую роль в обеспечении эффективности и гибкости ИТ-инфраструктуры, а важность обеспечения безопасности информационной инфраструктуры, построенной на виртуальных машинах (ВМ), становится непреложной. Несмотря на это, существует серьезная угроза для целостности и безопасности данных организаций в случае проблем с миграцией образов виртуальных машин из-за несовместимости аппаратного и программного обеспечения, например [1, 2].

За последние годы многими исследователями было предложено и реализовано значительное количество работ, касающихся безопасности технологий, связанных с виртуализацией, например [1–8]. Однако пробелом остается разработка универсальных

1 Корнеев Николай Владимирович, доктор технических наук, доцент, РГУ нефти и газа (НИУ) имени И. М. Губкина, Москва, Россия. E-mail: niccyper@mail.ru
2 Дикий Александр Борисович, студент РГУ нефти и газа (НИУ) имени И. М. Губкина, Москва, Россия. E-mail: wild.alex2016@yandex.ru

шаблонных механизмов безопасности, способных поддерживать процесс безопасной миграции образов виртуальных машин в информационной инфраструктуре, называемыми паттернами. В частности, в данной статье мы ставим целью разработку шаблонного механизма защиты для обеспечения безопасности информационной инфраструктуры при миграции образов виртуальных машин.

Паттерн безопасности описывает конкретную повторяющуюся проблему безопасности, которая возникает в определенных известных контекстах, а также предлагает хорошо зарекомендовавшую себя общую схему решения такой проблемы безопасности [9].

Область виртуализации информационной инфраструктуры представляет достаточно объемную область исследования, поэтому в данной статье мы ограничимся вопросом обеспечения безопасности при миграции образа виртуальной машины с одного устройства на другое. Для практического применения полученных результатов фокус статьи смещен на обеспечение безопасности на ранних стадиях процесса миграции, а именно рассматривается возможность совместимости хостов сред виртуализации, которым принадлежит образ ВМ. Это подразумевает наличие нескольких серверов с различными видами программного, аппаратного обеспечения и сред виртуализации. Для организации связи между серверами будет использоваться ssh (secure shell) протокол, как наиболее известный и широко применяемый сетевой протокол связи между ВМ, посредством которого они могут взаимодействовать и обмениваться данными. К каждой ВМ можно получить доступ при помощи ssh тунеля, который позволяет получить доступ к удалённой машине с любого устройства, на котором есть необходимые ssh ключи [9]. Передаваемые данные при этом шифруются, поэтому протокол ssh считается безопасным и подходит для передачи чувствительных данных.

Согласно банку угроз безопасности информации ФСТЭК России, такая угроза безопасности информации называется УБИ 052. Она может быть осуществлена внешним нарушителем с низким потенциалом. Угроза заключается в возникновении технических сложностей при миграции образов ВМ между поставщиками облачных услуг, обусловленных несовместимостью аппаратного и программного обеспечения.

Анализ и методы исследования

Система виртуализации представляет собой программное обеспечение, которое эмулирует аппаратное обеспечение, позволяя запускать несколько виртуальных машин на одном физическом сервере.

Данная система играет свою роль в безопасности – виртуализация создает изолированные среды (ВМ), что способствует снижению рисков и управлению безопасностью, так как компрометация одной виртуальной машины не влияет на другие.

При необходимости сменить сервер или систему виртуализации возможно использовать процесс миграции образа виртуальной машины для переноса данных. Существует несколько современных форматов миграции [10], в том числе холодная и горячая, отличающиеся состоянием ВМ. При холодной миграции необходимо отключить ВМ, при горячей – ВМ продолжает работу без значительного прерывания работы. Схематическое изображение процесса миграции образа ВМ представлено на (рис. 1).

Из рис. 1 видно, что процесс миграции образа виртуальной машины проходит этапы совместимости с программным обеспечением виртуализации – гипервизором и аппаратным обеспечением – сервером. Один из основных аспектов при миграции – это совместимость образов виртуальных машин

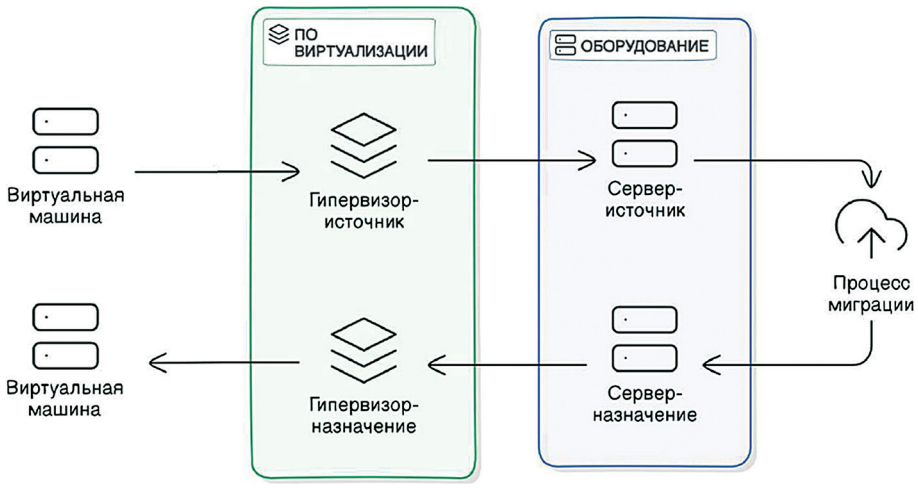

Рис. 1. Схема процесса миграции образа виртуальной машины

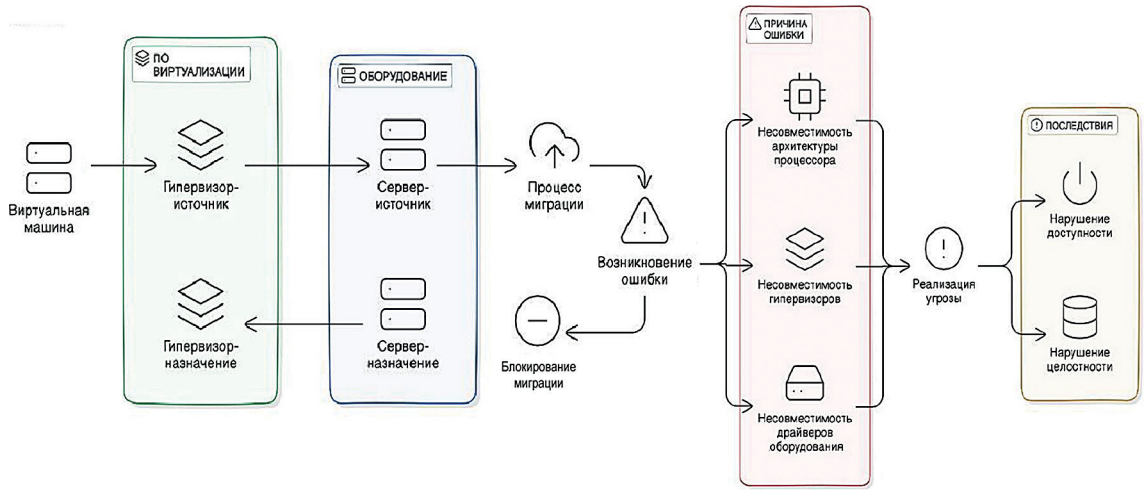

Рис. 2. Схема процесса миграции образа виртуальной машины с ошибками и последствиями реализации угрозы

с целевой платформой, что включает аппаратное (сервер назначения) и программное обеспечение (гипервизор назначения). Безопасное и успешное завершение миграции требует гарантии целостности данных в процессе перемещения образов виртуальных машин.

Невозможность миграции образов виртуальных машин из-за несовместимости аппаратного и программного обеспечения может порождать ряд серьезных ошибок (рис.2), связанных с нарушением целостности и доступности системы, например [11]. В качестве последствий реализации угрозы нами рассматриваются:

1. Нарушения целостности:
   1.1. Несовместимость между аппаратным и программным обеспечением может привести к повреждению или потере данных при миграции образа ВМ.
   1.2. Ошибки в миграции могут вызвать искажение конфигураций ВМ или потерю части данных, что приведёт к нарушению целостности самих информационных и программных систем, развернутых на ВМ.
2. Нарушения доступности:
   2.1. Отказ в обслуживании DoS (Denial of Service) [12] в случае невозможности миграции образа ВМ из-за несовместимости между аппаратным и программным обеспечением, при которой критически важные приложения или сервисы не смогут работать.
   2.2. В случае выхода из строя оборудования или необходимости его замены невозможность миграции образа ВМ может привести к простою и недоступности сервисов.

Для ограничения зоны применимости разрабатываемого шаблонного механизма защиты определим, что нами будет рассматриваться конкретный сценарий атаки (рис. 3). Атакующий – внешний нарушитель с низким потенциалом, согласно ФСТЭК. Нарушитель получил доступ к интерфейсам управления среды виртуализации или же самого сервера. Цель нарушителя – вызвать отказ в обслуживании (DoS) или же повреждение данных ВМ, для чего инициируется процесс миграции образа ВМ в несовместимую среду.

Для предотвращения атаки необходимо не допустить миграции образа ВМ в несовместимую среду. Для этого необходимо реализовать паттерн безопасности, включающий в себя механизм защиты информационной инфраструктуры при миграции образов виртуальных машин (рис. 4) в виде сервиса с решениями, способными из защищенной среды, не связанной с основным сервером, провести диагностику системы сервера-назначения и установленного гипервизора перед миграцией. В случае несовместимости, сервис должен остановить процесс миграции образа ВМ.

Механизм защиты строится на применении методов низкоуровневого программирования путем вызова, анализа системных функций через высокоуровневые системные библиотеки и формирования специального слоя системных функций для принятия окончательного решения о миграции образа ВМ. Специальный слой системных функций реализуется как сервис-решение в защищенной среде с использованием ssh протокола.

Разрабатываемый нами паттерн может быть использован не только для защиты описанной информационной инфраструктуры, но и в более широком контексте – для обеспечения безопасности информационной инфраструктуры в целом, состоящей из различного набора ВМ, гипервизоров и серверов, в этом случае предлагаемое решение может рассматриваться как кроссплатформенное.

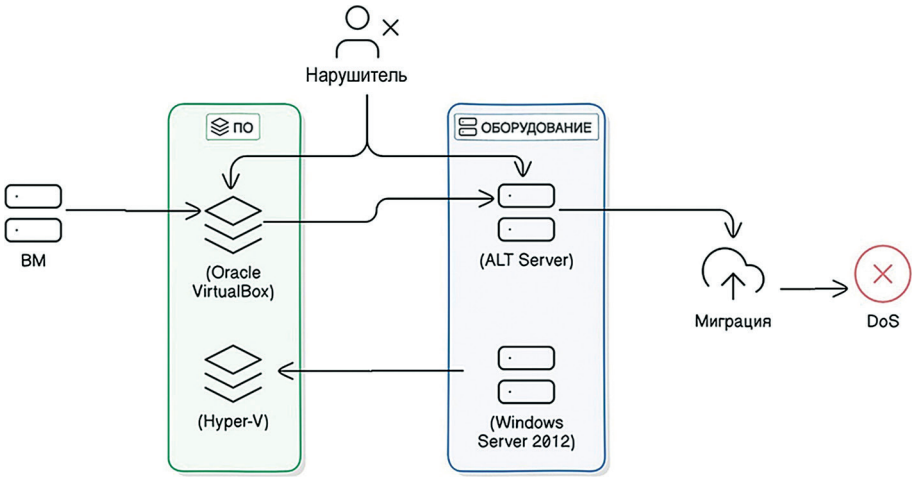

Рис. 3. Сценарий атаки отказ в обслуживании (DoS) в процессе миграции образа виртуальной машины в несовместимую среду

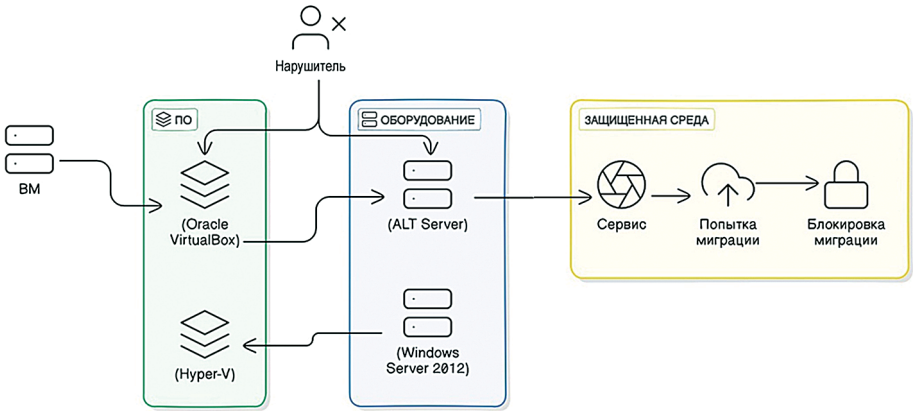

Рис. 4. Сценарий защиты информационной инфраструктуры при миграции образов виртуальных машин

Новизна предлагаемого решения определяется возможностью обеспечения безопасности информационной инфраструктуры при угрозе отказа в обслуживании (DoS) или же повреждении данных ВМ в процессе миграции образа ВМ в несовместимую среду в облачной информационной инфраструктуре России.

Практическая значимость предлагаемого решения включает шаблонный механизм защиты в виде паттерна, который можно применить для различной по составу информационной инфраструктуры (ВМ, гипервизоры и серверы), в том числе перенести разрабатываемое решение на любую отрасль: топливно-энергетическую, экономическую и не только, ввиду кроссплатформенности самого решения.

Паттерн безопасности для информационной инфраструктуры

Для реализации паттерна использован следующий стек технологий: Python и библиотеки subprocess, platform, sys, psutil, paramiko, Docker, Docker Engine, VirtualBox и специальные функции для проверки ВМ, Graphana и Prometheus.

Паттерн безопасности (рис. 5) построен на основе микросервисной архитектуры и представляет собой сервис-решение. Сама архитектура микросервисов (Microservice Architecture, MA) представляет собой новую парадигму архитектуры программного обеспечения, которая направлена на решение ограничений традиционного монолитного программного обеспечения путем разложения всего программного обеспечения на независимо развертываемые и масштабируемые более мелкие части, называемые сервисами или микросервисами [13]. За основу взят язык Python, виртуализация выполнена с использованием Docker и VirtualBox, а для диагностики и мониторинга информационной инфраструктуры использованы Graphana и Prometheus.

Ядро данного паттерна включает в себя механизм проверки целевого хоста на соответствие характеристикам и версиям основного программного

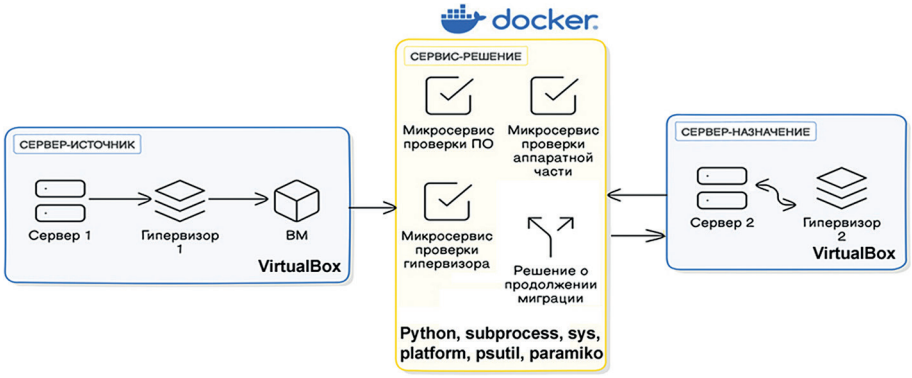

Рис. 5. Микросервисная архитектура паттерна безопасности информационной инфраструктуры при миграции образов виртуальных машин

и аппаратного обеспечения, используемого на текущем хосте виртуальной машины. Для этого реализуется специальный слой системных функций, обеспечивающих сбор и анализ информации о целевой и текущей системах, включая не только технические параметры, но и данные о версиях операционной системы, драйверов и другого ключевого программного обеспечения. Специальный слой системных функций реализован микросервисами:

1. Микросервис проверки ПО осуществляет сбор данных о совместимости. Микросервис реализует один из ключевых аспектов системных характеристик – это процесс аккумулирования данных о совместимости между текущей ВМ и потенциальным целевым хостом. Это включает в себя сведения о версиях операционных систем, требуемых плагинах, драйверах и других компонентах, необходимых для корректной работы приложения.
2. Микросервис проверки аппаратной части осуществляет оценку технических характеристик для обеспечения эффективной миграции образа ВМ. Заложенные в него системные функции включают в себя анализ технических параметров, таких как архитектура процессора, доступные объемы памяти и хранилища, а также наличие специфических функциональных возможностей, необходимых для работы информационной инфраструктуры.
3. Микросервис проверки гипервизора определяет стабильность и надежность целевого хоста. Помимо соответствия ВМ программному обеспечению гипервизора, он также включает методы проверки стабильности и надежности целевой системы. Это важно для предотвращения потенциальных сбоев или уязвимостей, которые могут возникнуть в результате миграции образа ВМ на несовместимый хост.

Для практической реализации паттерна на любой информационной инфраструктуре потребуются знания о целевом хосте, на который планируется миграция образа ВМ. Прежде всего это ip-адрес устройства, поскольку используется соединение серверов по ssh протоколу. Помимо этого, необходимо наличие любой среды виртуализации, заранее установленной на целевом устройстве.

Для принятия окончательного решения о миграции образа ВМ используется разработанный нами специальный слой системных функции:

- набор функций проверки версии VirtualBox производит проверку версии VirtualBox на удаленном хосте;
- функция проверки ресурсов системы проверяет доступную память и количество ядер процессора;
- набор функций проверки совместимости файловой системы и доступного места на диске осуществляет проверку совместимости файловой системы между локальным и удаленным хостами и доступного места на жестком диске удаленного хоста;
- функция проверки совместимости оборудования и ресурсов проверяет совместимость оборудования и ресурсов текущего хоста с удаленным хостом;
- функция проведения всех проверок и журналирования результатов проводит проверку по всем функциям и сохраняет результаты проверки в отдельный файл.

Разработка специального слоя системных функции

Программная часть паттерна и специальный слой системных функций реализован на языке Python. Выбор обусловлен простым синтаксисом языка Python и его гибкостью, благодаря большому количеству встроенных библиотек для работы с системными переменными сред. В коде были использованы следующие библиотеки:

1. Python subprocess использована для выполнения команд VBoxManage, для получения информации о виртуальных машинах VirtualBox, версии гипервизора, а также сетевых интерфейсов

и процессора. Эта библиотека позволяет запускать системные команды и получать их вывод, что важно для проверки совместимости аппаратного и программного обеспечения.

2. Python platform использована для определения операционной системы, чтобы правильно адаптировать команды для получения информации об информационной инфраструктуре, а также учесть, что для разных операционных систем команды будут различные.
3. Python sys использована для завершения программы с помощью функции sys.exit(), если одна из проверок с помощью специального слоя системных функций не пройдена. Это позволяет прервать выполнение скрипта, чтобы предотвратить миграцию ВМ в случае выявления несовместимости.
4. Psutil [14, 15] использована для доступа к информации о системе и процессах из кода на языке Python. Это позволяет проводить мониторинг различных системных ресурсов, таких как CPU, память, диски, сетевые интерфейсы и другие ресурсы.
5. Paramiko [16, 17] использована для работы с протоколами SSH, SFTP, SCP из кода на языке Python, что обеспечило паттерн сетевыми возможностями для безопасного удаленного управления и передачи данных в информационной инфраструктуре.

Разработанный нами специальный слой системных функции включает:

1. Набор функций проверки версии VirtualBox:

```
def execute_vboxmanage_command(command);
def check_virtualbox_version();
def check_target_virtualbox_version().
```

2. Функцию проверки ресурсов систем:

```
def check_hardware_resources().
```

3. Набор функций проверки совместимости файловой системы и доступного места на диске:

```
def check_remote_filesystem_compatibility();
def check_remote_disk_space();
def additional_checks_remote().
```

4. Функцию проверки совместимости оборудования и ресурсов:

```
def check_host_compatibility().
```

5. Функцию проведения всех проверок и журналирования результатов:

```
def additional_checks().
```

Далее приведем примеры нескольких разработанных нами функций из специального слоя системных функций.

Код функции execute_vboxmanage_command(command):

```
def execute_vboxmanage_command(command):
   result = subprocess.run(['VBoxManage'] +
command.split(), stdout=subprocess.PIPE)
   return result.stdout.decode('utf-8')
```

Эта функция выполняет команды VBoxManage для управления VirtualBox. Параметры функции – строка команды для выполнения. Возвращаемое значение функции – результат выполнения команды в формате строки.

Код функции additional_checks():

```
   def additional_checks():
   error_messages=[]
   functions=[check_virtualbox_version,
check_target_virtualbox_version, check_
hardware_resources, check_host_compatibility]
   for func in functions:
   if func==check_target_virtualbox_version:
   success, message=func(target_host_ip,
username, password)
   else:
   success,message=func()
   if not success:
   error_messages.append(message)
   remote_errors=additional_checks_remote
(target_host_ip, username, password)
   error_messages.extend(remote_errors)
   return error_messages
   target_host_ip=input("Введите IP-адрес
удаленного хоста:")
   username=input("Введите имя пользователя
для удаленного хоста:")
   password=input("Введите пароль для удален-
ного хоста:")
   output_file="security_check_results.txt"
   error_messages=additional_checks()
   if error_messages:
   print("Ошибки записаны в журнал проверок
security_check_results")
   with open(output_file, 'a') as file:
   for error in error_messages:
   file.write("- " + error + "\n")
   else:
   print("Проверки завершены без ошибок")
```

Эта функция выполняет все проверки на удаленном хосте и сохраняет результаты проверки в отдельный файл. В коде с помощью первого цикла for и условного оператора if проводится проверка передачи в функции правильного набора аргументов: target_host_ip – ip-адрес устройства, username – логин, password – пароль. С помощью методов append() и extend() добавляются сообщения об ошибках

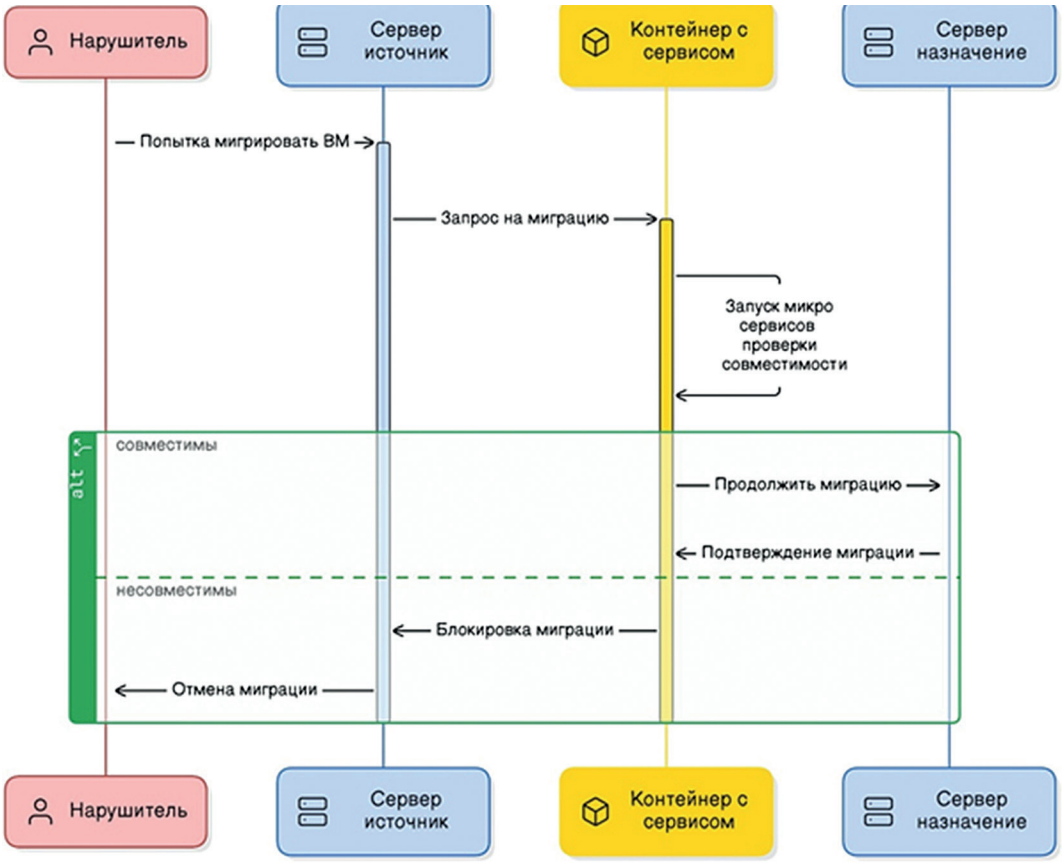

Рис. 6. UML-диаграмма последовательности паттерна безопасности

на удаленном хосте к общему списку ошибок. Далее в коде у пользователя запрашиваются необходимые данные для проведения проверок: target_host_ip, username, password. Затем проводятся проверки с использованием специального слоя системных функции. Результаты проверок записываются в файл security_check_results.txt.

Разработка паттерна безопасности

Паттерн безопасности реализован в виде Docker-контейнера. Нами использованы объявленные ранее микросервисы для выполнения различных проверок и журналирования результатов. Рассмотрим диаграммы классов и взаимодействия, а также создадим Dockerfile для сборки контейнера. Dockerfile – это текстовый файл, который содержит инструкции для создания Docker-образа.

На (рис. 6) изображена UML-диаграмма последовательности для разработанного паттерна безопасности информационной инфраструктуры при миграции образов виртуальных машин.

Центральная роль на диаграмме (рис. 6) отведена контейнеру на базе Docker с предложенным нами выше сервис-решением. Микросервисы, входящие в состав контейнера, осуществляют проверку совместимости ВМ с использованием специального слоя системных функций и логируют ошибки при их возникновении с помощью специальных классов (рис. 7). По результатам проверки и анализа логов принимается решение о продолжении или блокировке миграции ВМ.

Рис. 7. Диаграмма состояний микросервисов паттерна безопасности

Диаграмма состояний микросервисов паттерна безопасности информационной инфраструктуры (рис. 7) показывает состояния и переходы между специальными классами: CheckSoftware, CheckHardware, CheckHypervisor, связанные с соответствующими микросервисами проверки ПО, проверки аппаратной части, проверки гипервизора.

Эти классы логируют ошибки при миграции образа ВМ. Класс Logs используется механизмом защиты для принятия решения об остановке или продолжении миграции.

Далее рассмотрим непосредственно процесс создания Docker-контейнера. Для его создания необходимо написать специальный текстовый файл Dockerfile. Dockerfile представляет собой набор команд, которые Docker Engine выполняет последовательно, чтобы собрать образ контейнера. Эту последовательность далее представим в виде алгоритма:

1. Загружаем официальный образ Python командой:

```
from python:3.9-slim
```

2. Устанавливаем зависимости для установки VirtualBox и библиотеки Python paramiko, выполняя команды:

```
run apt-get update && apt-get install -y
\wget
\gnupg2
\lsb-release
\iproute2
\sudo
\build-essential
\libffi-dev
\libssl-dev
\&& rm -rf /var/lib/apt/lists/*
```

3. Добавляем ключ репозитория Oracle для VirtualBox и обновляем репозитории, выполняя команды:

```
run wget -q https://www.virtualbox.org/
download/oracle_vbox_2016.asc -O- | apt-key
add - &&\
wget -q https://www.virtualbox.org/
download/oracle_vbox.asc -O- | apt-key add -
&&\
echo "deb [arch=amd64] http://download.
virtualbox.org/virtualbox/debian $(lsb_
release -cs) contrib"> /etc/apt/sources.
list.d/virtualbox.list &&\
apt-get update
```

4. Устанавливаем VirtualBox, выполняя команды:

```
run apt-get install -y\
virtualbox-6.1\
&& rm -rf /var/lib/apt/lists/*
```

5. Устанавливаем библиотеку Python paramiko, выполнив команду:

```
run pip install paramiko
```

6. Добавляем пользователя для VirtualBox, выполняя команды:

```
run useradd -ms /bin/bash vboxuser
run adduser vboxuser sudo
run echo 'vboxuser ALL=(ALL) NOPASSWD:ALL'
>> /etc/sudoers
```

7. Добавляем Python скрипт в контейнер, выполняя команды:

```
workdir /app
copy ./app
```

8. Устанавливаем Python зависимости, используя файл requirements.txt, выполнив команду:

```
run pip install -r requirements.txt
```

Данный пункт алгоритма можно пропустить, если нет Python зависимостей.

9. Делаем Python скрипт исполняемым, выполнив команду:

```
run chmod +x /app/kurs2.py
```

10. Для запуска контейнера от имени созданного пользователя выполнить команду:

```
user vboxuser
```

11. Для запуска Python скрипта выполнить команду:

```
CMD ["python3", "/app/kurs2.py"]
```

Таким образом, выполнив вышеописанный алгоритм, получаем Dockerfile и в той же директории собираем Docker-образ с именем vbox-checker командой:

```
docker build -t vbox-checker
```

Процесс создания Docker-образа выполняется согласно типовым инструкциям настройки и конфигурирования [14, 18].

Тестирование и мониторинг Docker-контейнера и обсуждение результатов

Для проведения тестирования и мониторинга необходимы инструменты сбора данных с ВМ и контейнера, а также инструмент визуализации полученных данных. В качестве таких инструментов нами использованы: Prometheus и Grafana [19], а также cAdvisor и Node Exporter [20].

Для наглядности тестирования далее будем рассматривать сценарий локального подключения серверов посредством ssh соединения, а собираемые метрики будут использованы для анализа различных параметров подключения. Для снятия метрик был использовано расширение prometheus-ssh-exporter инструмента Prometheus, способное анализировать ssh трафик.

Далее приведем результаты запуска собранного ранее контейнера и его тестирования. На (рис. 8) изображена консоль вывода с результатами запуска контейнера при несовместимости версии гипервизора. Из (рис. 8) видно, что причина остановки миграции образа ВМ – различие в версиях гипервизора.

```
host-15 kurs # docker run --network="host" --rm --name vbox-checker     -v /dev/
vboxdrv:/dev/vboxdrv     --privileged     vbox-checker
Версия VirtualBox на исходном хосте: 6.1.50r161033
Версия VirtualBox на целевом хосте: 6.1.40r154048
Версии VirtualBox не совпадают, возможны проблемы при миграции.
Миграция остановлена: несовместимые версии VirtualBox
```

Рис. 8. Консоль вывода с результатами запуска контейнера при несовместимости версии гипервизора

Приведем результаты тестирования созданных ранее микросервисов паттерна безопасности. В результате запуска контейнера и Python скрипта, согласно п. 11 алгоритма, были получены следующие данные, изображенные на (рис. 9).

```
[root@9ddb5e669bee app]# python3 kurs2.py
Введите IP-адрес удаленного хоста: 192.168.56.3
Введите имя пользователя для удаленного хоста: user
Введите пароль для удаленного хоста: resu
Версия VirtualBox на удаленном хосте: 6.1.50r161033
Доступная память: 2110615552 байт
Количество ядер CPU: 3
Ошибки записаны в журнал проверок security_check_results
[root@9ddb5e669bee app]# ls
kurs2.py  security_check_results.txt
[root@9ddb5e669bee app]#
```

Рис. 9. Консоль вывода с результатами работы микросервисов

Из (рис. 9) видно, что соответствующими микросервисами были выполнены все предусмотренные нами проверки: ПО, аппаратной части, проверки гипервизора, а результат неудачной попытки миграции образа ВМ записан в файл логов security_check_results.txt (рис. 10). В нем же содержаться данные о причинах неудачной попытки миграции образа ВМ. Содержимое файла изображено на (рис. 10).

```
[root@9ddb5e669bee app]# cat security_check_results.txt
Результаты проверки безопасности удаленного хоста:
- Недостаточно оперативной памяти (3.84 из '4 Gb')
- Несовместимая файловая система с хостом (Файловая система: overlay)
- Недостаточно места на диске. Требуется как минимум 40 ГБ, доступно: 29 ГБ
[root@9ddb5e669bee app]#
```

Рис. 10. Консоль вывода содержания файла security_check_results.txt

Далее приведем пример визуализации метрик, взятых посредством инструментов мониторинга Graphana и Prometheus. После загрузки указанных выше инструментов, необходимо перейти на сайт Grafana посредством ввода в адресную строку адреса, включающего в себя ip-адрес текущего устройства с портом 3000. На (рис. 11) изображена витрина Graphana с количеством ssh подключений, выполненных за период работы контейнера.

Рис. 11. Витрина Graphana с количеством ssh подключений

Из (рис. 11) видно, что первое и второе подключение по ip-адресу 172.18.176.1 прошло успешно, и можно сделать вывод о начале миграции образа ВМ. Однако для более корректного вывода следует настроить метрики ошибок более гибко через соответствующие события, связанные со специальными классами: CheckSoftware, CheckHardware, CheckHypervisor. Третье подключение по ip-адресу 172.18.176.1 было неудачным, и можно сделать вывод о блокировании попытки миграции образа ВМ классом Logs.

Выводы

Построена микросервисная архитектура для обеспечения безопасности информационной инфраструктуры при миграции образов виртуальных машин. Рассмотрен сценарий атаки отказ в обслуживании (DoS) в процессе миграции образа виртуальной машины в несовместимую среду. Разработан паттерн безопасности информационной инфраструктуры при миграции образов виртуальных машин на основе микросервисов, интегрированных в Docker-контейнер. Разработаны 3 микросервиса: проверки ПО; проверки аппаратной части; проверки гипервизора. Разработан специальный слой системных функций для принятия окончательного решения о миграции образа ВМ, включающий: набор функций проверки версии VirtualBox; функцию проверки ресурсов системы; набор функций проверки совместимости файловой системы и доступного места на диске; функцию проверки совместимости оборудования и ресурсов; функцию проведения всех проверок и журналирования результатов. Механизм защиты построен на применении методов низкоуровневого программирования путем вызова, анализа системных функций через высокоуровневые системные библиотеки и формирования специального слоя системных функций для принятия окончательного решения о миграции образа ВМ. Специальный слой системных функций реализуется как сервис-решение в защищенной среде с использованием ssh протокола. Разработан программный код микросервисов, включая коды для системных функции и специальных классов: CheckSoftware, CheckHardware, CheckHypervisor, Logs, обеспечивающих механизм защиты. Разработан алгоритм создания паттерна безопасности в виде Docker-контейнера, написан Dockerfile, создан Docker-образ и Docker-контейнер. Проведено тестирование микросервисов, которое показало успешную работу механизма защиты. Развёрнута система мониторинга процесса миграции образа ВМ в несовместимую среду на базе открытого ПО и созданного Docker-контейнера, которая может быть использована в системах SIEM, а метрики ошибок миграции могут быть настроены через соответствующие события специальных классов, что дает возможность гибко настраивать сам паттерн и применять его для широкого круга приложений.

Литература

1. Matheus Torquato, Paulo Maciel, Marco Vieira, Evaluation of time-based virtual machine migration as moving target defense against host-based attacks, Journal of Systems and Software, Volume 219, 2025, 112222. DOI:10.1016/j.jss.2024.112222.
2. Добродеев А. Ю. Кибербезопасность в Российской Федерации. Модный термин или приоритетное технологическое направление обеспечения национальной и международной безопасности XXI века // Вопросы кибербезопасности. 2021. №4 (44). С. 61–72. DOI: 10.21681/2311-3456-2021-4-61-72.
3. Chunjing Liu, Lixiang Ma, Minfeng Zhang, Haiyan Long, Optimizing cloud resource management with an IoT-enabled optimized virtual machine migration scheme for improved efficiency, Journal of Network and Computer Applications, Volume 237, 2025, 104137. DOI: 10.1016/j.jnca.2025.104137.
4. Hanif Deylami, Jairo Gutierrez, Roopak Sinha, Kororā: A secure live virtual machine job migration framework for cloud systems integrity, Array, Volume 19, 2023, 100312. DOI: 10.1016/j.array.2023.100312.
5. Rahmat Zolfaghari, Energy-performance aware virtual machines migration in cloud network by using prediction and fuzzy approaches, Engineering Applications of Artificial Intelligence, Volume 131, 2024, 107825. DOI: 10.1016/j.engappai.2023.107825.
6. Tahir Alyas, Taher M. Ghazal, Badria Sulaiman Alfurhood, Munir Ahmad, Ossma Ali Thawabeh, Khalid Alissa, Qaiser Abbas, Performance Framework for Virtual Machine Migration in Cloud Computing, Computers, Materials and Continua, Volume 74, Issue 3, 2022, 6289–6305. DOI: 10.32604/cmc.2023.035161.
7. Шелухин О. И., Раковский Д. И. Разработка программно-аппаратного комплекса моделирования многозначных компьютерных атак // Вопросы кибербезопасности. 2024. №4 (62). С. 116–130. DOI: 10.21681/2311-3456-2024-4-116-130.
8. Маркин Д. О., Макеев С. М. Система защиты терминальных программ от анализа на основе виртуализации исполняемого кода // Вопросы кибербезопасности. 2020. №1 (35). С. 29–41. DOI:10.21681/2311-3456-2020-01-29-41.
9. Корнеев Н. В., Котрини Е. С. Паттерн для обеспечения безопасности приложения при угрозе модификации модели машинного обучения // Вопросы кибербезопасности. 2025. №1 (65). С. 117–127. DOI: 10.21681/2311-3456-2025-1-117-127.
10. Hui Zhao, Nanzhi Feng, Jianhua Li, Guobin Zhang, Jing Wang, Quan Wang, Bo Wan, VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment, Journal of Parallel and Distributed Computing, Volume 176, 2023, 17–27. DOI: 10.1016/j.jpdc.2023.02.003.
11. Chawki EL BALMANY, Zakariae TBATOU, Ahmed ASIMI, Mohamed BAMAROUF, Secure Virtual Machine Image Storage Process into a Trusted Zone-based Cloud Storage, Computers & Security, Volume 120, 2022, 102815. DOI: 10.1016/j.cose.2022.102815.
12. Fargana J. Abdullayeva, Distributed denial of service attack detection in E-government cloud via data clustering, Array, Volume 15, 2022, 100229. DOI:10.1016/j.array.2022.100229.
13. Корнеев Н. В., Лазорин Д. С. Паттерн для обеспечения безопасности веб-приложения при угрозе XSS атак в облачной инфраструктуре // Вопросы кибербезопасности. 2024. №6 (64). С. 76–84. DOI: 10.21681/2311-3456-2024-6-76-84.
14. Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576. DOI: 10.1016/j.softx.2023.101576.
15. Khawaja Tahir Mehmood, Shahid Atiq, Intisar Ali Sajjad, Muhammad Majid Hussain, Malik M. Abdul Basit, Examining the Quality Metrics of a Communication Network with Distributed Software-Defined Networking Architecture, CMES - Computer Modeling in Engineering and Sciences, Volume 141, Issue 2, 2024, 1673–1708. DOI: 10.32604/cmes.2024.053903.
16. Derek Groen, Hamid Arabnejad, Diana Suleimenova, Wouter Edeling, Erwan Raffin, Yani Xue, Kevin Bronik, Nicolas Monnier, Peter V. Coveney, FabSim3: An automation toolkit for verified simulations using high performance computing, Computer Physics Communications, Volume 283, 2023, 108596. DOI: 10.1016/j.cpc.2022.108596.
17. Sebastian Troia, Marco Savi, Giulia Nava, Ligia Maria Moreira Zorello, Thomas Schneider, Guido Maier, Performance characterization and profiling of chained CPU-bound Virtual Network Functions, Computer Networks, Volume 231, 2023, 109815. DOI:10.1016/j.comnet.2023.109815.
18. Hubin Yang, Ruochen Shao, Yanbo Cheng, Yucong Chen, Rui Zhou, Gang Liu, Guoqi Xie, Qingguo Zhou, REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems, Journal of Systems Architecture, Volume 151, 2024, 103135. DOI:10.1016/j.sysarc.2024.103135.
19. Vladimir Ciric, Marija Milosevic, Danijel Sokolovic, Ivan Milentijevic, Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation, Simulation Modelling Practice and Theory, Volume 133, 2024, 102916. DOI: 10.1016/j.simpat.2024.102916.
20. Miguel Correia, Wellington Oliveira, José Cecílio, Monintainer: An orchestration-independent extensible container-based monitoring solution for large clusters, Journal of Systems Architecture, Volume 145, 2023, 103035. DOI:10.1016/j.sysarc.2023.103035.

PATTERN FOR SECURING INFORMATION INFRASTRUCTURE DURING VIRTUAL MACHINE IMAGE MIGRATION

Korneev N. V.[3], Dikiy A. B.[4]

Keywords: *cloud computing, virtualization, template, denial of service attack, incompatible environment, special system function layer, ssh protocol, hypervisor, server, container, event log, monitoring system.*

3 Nikolai Korneev, Dr.Sc., professor, Gubkin Russian State University of Oil and Gas (National Research University), Moscow. E-mail: niccyper@mail.ru
4 Dikiy Alexander, student, Gubkin Russian State University of Oil and Gas (National Research University), Moscow. E-mail: wild.alex2016@yandex.ru

The purpose of this article: *development of a template protection mechanism to ensure the security of the information infrastructure during the migration of virtual machine images.*

Research method: *analysis of hardware virtualization principles and the process of virtual machine image migration. Synthesis of a denial of service (DoS) attack scenario during the process of virtual machine image migration to an incompatible environment. Using low-level programming methods, a new protection mechanism is proposed by calling, analyzing system functions through high-level system libraries and forming a special layer of system functions to make a final decision on virtual machine image migration. The special layer of system functions is implemented as a service solution in a secure environment using the ssh protocol. The study was carried out by full-scale modeling of the Docker-based information infrastructure in environments with containerization support, its deployment and testing during the implementation of the attack scenario.*

Result: *the analysis of the threat of technical difficulties during migration of virtual machine images between cloud service providers caused by incompatibility of hardware and software is carried out, and the relevance of the problem of developing universal template security mechanisms, called patterns, is shown. A microservice architecture is built to ensure the security of the information infrastructure during migration of virtual machine images. A denial of service (DoS) attack scenario is considered during migration of a virtual machine image to an incompatible environment. A security pattern of the information infrastructure during migration of virtual machine images is developed based on microservices integrated into a Docker container. Three microservices are developed: software checks; hardware checks; hypervisor checks. A special layer of system functions has been developed to make the final decision on virtual machine image migration, including: a set of functions for checking the VirtualBox version; a function for checking system resources; a set of functions for checking file system compatibility and available disk space; a function for checking hardware and resource compatibility; a function for performing all checks and logging the results. The program code of microservices has been developed, including codes for system functions and special classes: CheckSoftware, CheckHardware, CheckHypervisor, Logs providing a protection mechanism. An algorithm for creating a security pattern in the form of a Docker container has been developed, a Dockerfile has been written, a Docker image and a Docker container have been created. Microservices have been tested, which has shown the successful operation of the protection mechanism. A system for monitoring the process of migrating a virtual machine image to an incompatible environment based on open source software and the created Docker container has been deployed, which can be used in SIEM systems, and migration error metrics can be configured through the corresponding events of special classes, which makes it possible to flexibly configure the pattern itself and apply it to various information infrastructures.*

Practical value: *the practical value of the proposed solution includes a template protection mechanism in the form of a pattern that can be used to ensure the security of various information infrastructures consisting of a different set of VMs, hypervisors and servers, including transferring the developed solution to any industry: fuel and energy, economic and not only, due to the cross-platform nature of the solution itself.*

References

1. Matheus Torquato, Paulo Maciel, Marco Vieira, Evaluation of time-based virtual machine migration as moving target defense against host-based attacks, Journal of Systems and Software, Volume 219, 2025, 112222. DOI:10.1016/j.jss.2024.112222.
2. Dobrodeev A. Yu. Kiberbezopasnost' v Rossijskoj Federacii. Modny'j termin ili prioritetnoe texnologicheskoe napravlenie obespecheniya nacional'noj i mezhdunarodnoj bezopasnosti XXI veka // Voprosy kiberbezopasnosti. 2021. №4 (44). S. 61–72. DOI: 10.21681/2311-3456-2021-4-61-72.
3. Chunjing Liu, Lixiang Ma, Minfeng Zhang, Haiyan Long, Optimizing cloud resource management with an IoT-enabled optimized virtual machine migration scheme for improved efficiency, Journal of Network and Computer Applications, Volume 237, 2025, 104137. DOI: 10.1016/j.jnca.2025.104137.
4. Hanif Deylami, Jairo Gutierrez, Roopak Sinha, Kororā: A secure live virtual machine job migration framework for cloud systems integrity, Array, Volume 19, 2023, 100312. DOI: 10.1016/j.array.2023.100312.
5. Rahmat Zolfaghari, Energy-performance aware virtual machines migration in cloud network by using prediction and fuzzy approaches, Engineering Applications of Artificial Intelligence, Volume 131, 2024, 107825. DOI: 10.1016/j.engappai.2023.107825.
6. Tahir Alyas, Taher M. Ghazal, Badria Sulaiman Alfurhood, Munir Ahmad, Ossma Ali Thawabeh, Khalid Alissa, Qaiser Abbas, Performance Framework for Virtual Machine Migration in Cloud Computing, Computers, Materials and Continua, Volume 74, Issue 3, 2022, 6289–6305. DOI: 10.32604/cmc.2023.035161.
7. Sheloukhin O.I., Rakovsky D.I. Razrabotka programmno-apparatnogo kompleksa modelirovaniya mnogoznachny'x komp'yuterny'x atak // Voprosy kiberbezopasnosti. 2024. №4 (62). S. 116–130. DOI: 10.21681/2311-3456-2024-4-116-130.
8. Markin D.O., Makeev S.M. Sistema zashhity' terminal'ny'x programm ot analiza na osnove virtualizacii ispolnyaemogo koda // Voprosy kiberbezopasnosti. 2020. №1 (35). S. 29–41. DOI:10.21681/2311-3456-2020-01-29-41.
9. Korneev N.V., Kotrini E.S. Pattern dlya obespecheniya bezopasnosti prilozheniya pri ugroze modifikacii modeli mashinnogo obucheniya // Voprosy kiberbezopasnosti. 2025. №1 (65). S. 117–127. DOI: 10.21681/2311-3456-2025-1-117-127.
10. Hui Zhao, Nanzhi Feng, Jianhua Li, Guobin Zhang, Jing Wang, Quan Wang, Bo Wan, VM performance-aware virtual machine migration method based on ant colony optimization in cloud environment, Journal of Parallel and Distributed Computing, Volume 176, 2023, 17–27. DOI: 10.1016/j.jpdc.2023.02.003.
11. Chawki EL BALMANY, Zakariae TBATOU, Ahmed ASIMI, Mohamed BAMAROUF, Secure Virtual Machine Image Storage Process into a Trusted Zone-based Cloud Storage, Computers & Security, Volume 120, 2022, 102815. DOI: 10.1016/j.cose.2022.102815.

12. Fargana J. Abdullayeva, Distributed denial of service attack detection in E-government cloud via data clustering, Array, Volume 15, 2022, 100229. DOI:10.1016/j.array.2022.100229.
13. Korneev N.V., Lazorin D.S. Pattern dlya obespecheniya bezopasnosti veb-prilozheniya pri ugroze XSS atak v oblachnoj infrastrukture // Voprosy kiberbezopasnosti. 2024. №6 (64). S. 76–84. DOI: 10.21681/2311-3456-2024-6-76-84.
14. Enrico Cambiaso, Luca Caviglione, Marco Zuppelli, DockerChannel: A framework for evaluating information leakages of Docker containers, SoftwareX, Volume 24, 2023, 101576. DOI: 10.1016/j.softx.2023.101576.
15. Khawaja Tahir Mehmood, Shahid Atiq, Intisar Ali Sajjad, Muhammad Majid Hussain, Malik M. Abdul Basit, Examining the Quality Metrics of a Communication Network with Distributed Software-Defined Networking Architecture, CMES - Computer Modeling in Engineering and Sciences, Volume 141, Issue 2, 2024, 1673–1708. DOI: 10.32604/cmes.2024.053903.
16. Derek Groen, Hamid Arabnejad, Diana Suleimenova, Wouter Edeling, Erwan Raffin, Yani Xue, Kevin Bronik, Nicolas Monnier, Peter V. Coveney, FabSim3: An automation toolkit for verified simulations using high performance computing, Computer Physics Communications, Volume 283, 2023, 108596. DOI: 10.1016/j.cpc.2022.108596.
17. Sebastian Troia, Marco Savi, Giulia Nava, Ligia Maria Moreira Zorello, Thomas Schneider, Guido Maier, Performance characterization and profiling of chained CPU-bound Virtual Network Functions, Computer Networks, Volume 231, 2023, 109815. DOI:10.1016/j.comnet.2023.109815.
18. Hubin Yang, Ruochen Shao, Yanbo Cheng, Yucong Chen, Rui Zhou, Gang Liu, Guoqi Xie, Qingguo Zhou, REDB: Real-time enhancement of Docker containers via memory bank partitioning in multicore systems, Journal of Systems Architecture, Volume 151, 2024, 103135. DOI:10.1016/j.sysarc.2024.103135.
19. Vladimir Ciric, Marija Milosevic, Danijel Sokolovic, Ivan Milentijevic, Modular deep learning-based network intrusion detection architecture for real-world cyber-attack simulation, Simulation Modelling Practice and Theory, Volume 133, 2024, 102916. DOI: 10.1016/j.simpat.2024.102916.
20. Miguel Correia, Wellington Oliveira, José Cecílio, Monintainer: An orchestration-independent extensible container-based monitoring solution for large clusters, Journal of Systems Architecture, Volume 145, 2023, 103035. DOI:10.1016/j.sysarc.2023.103035.