# АННОТАЦИЯ

Тема выпускной квалификационной работы магистра — «Исследование процессов обеспечения безопасности облачных сред».

Ключевые слова: безопасность, виртуализация, защита информации, облачная инфраструктура, облачные вычисления, провайдер, стандартизация, уязвимости.

В данной выпускной квалификационной работе магистра рассмотрены проблемы обеспечения безопасности облачных сред, а также представлены примеры решения этих проблем. Для достижения поставленной цели был необходим детальный анализ работы облачных провайдеров, мировых стандартов безопасности для поставщиков облачных услуг, а также технические возможности эксплуатации уязвимостей в облачной среде на разных уровнях, с целью компрометации данных пользователей или данных облачного провайдера.

Актуальность темы. В настоящее время наблюдается стремительное развитие облачных технологий, однако для эффективной работы облачной инфраструктуры требуется ее эффективная структура и организация. Зачастую небольшая команда, проектирующая облачную инфраструктуру не всегда может полностью учесть все аспекты безопасности, так как нет единого документа, стандартизующего механизмы обеспечения безопасности в облачной среде. Особенно остро вопрос безопасности встал в последнее время, в связи с обнаружением большого количества критических уязвимостей в программном обеспечении и протоколах, используемых поставщиками облачных услуг.

Целью проводимых исследований является повышение эффективности процессов обеспечения информационной безопасности облачных сред. Для достижения поставленной цели необходимо разработать структуру системы обеспечения информационной безопасности заданной облачной среды, провести внедрение и экспериментальные исследования.

Выпускная квалификационная работа магистра изложена на 90 листах, включает 16 таблиц, 11 рисунков, 1 приложение, 28 литературных источников.

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Облачные услуги — это способ предоставления, потребления и управления технологией. Данный тип услуг выводит гибкость и эффективность на новый уровень, путем эволюции способов управления, таких как непрерывность, безопасность, резервирование и самообслуживание, которые соединяют физическую и виртуальную среду.

Для эффективной работы облачной инфраструктуры требуется эффективная структура и организация. Небольшая команда из специалистов и бизнес-пользователей может создать обоснованный план и организовать свою работу в подобной инфраструктуре. Данная выделенная группа может намного эффективнее построить и управлять нестандартной облачной инфраструктурой, чем если компании будут просто продолжать добавлять дополнительные сервера и сервисы для поддержки центра обработки данных (ЦОД).

Развитие информационного мира движется в сторону повсеместного распространения облачных вычислений, их технологий и сервисов. Очевидные преимущества данного подхода [1]:

- снижение затрат — отсутствие необходимости покупки собственного оборудования, программного обеспечения (ПО), работы системного инженера;
- удаленный доступ — возможность доступа к данным облака из любой точки мира, где есть доступ в глобальную сеть Интернет;
- отказоустойчивость и масштабируемость — изменение необходимых ресурсов в зависимости от потребностей проекта, техническое обслуживание оборудования лежит на плечах облачного провайдера.

В связи с этим можно сделать вывод, что основные недостатки облачных вычислений сводятся к информационной безопасности. Такого мнения придерживаются многие крупные информационные компании, что в некоторой степени препятствует более стремительному развитию рынка облачных сервисов.

# 1 ПОСТАНОВКА ЗАДАЧИ

Конечной задачей выпускной квалификационной работы магистра на тему «Исследование процессов обеспечения безопасности облачных сред» является повышение эффективности процессов обеспечения информационной безопасности облачных сред. Для достижения поставленной цели необходимо разработать структуру системы обеспечения информационной безопасности заданной облачной среды, провести внедрение и экспериментальные исследования.

Исследования должны состоять из следующих частей:

- обзор составных частей облачной инфраструктуры;
- анализ технологий используемых облачными провайдерами, необходимых для построения облачной инфраструктуры;
- специфика применений облачных вычислений в России;
- исследование проблемы безопасности облачных вычислений;
- решение проблем безопасности облаков;
- практическое применение уязвимостей в облачной среде, с использованием программ, распространяющихся под свободными лицензиями, например GNU GPL.

Для применения практических навыков исследования уязвимостей необходима аппаратная платформа со следующими характеристиками:

- процессор с поддержкой аппаратной виртуализации;
- минимальный объем ОЗУ 8 Гб, рекомендуемый — не менее 16 Гб;
- минимум 15 Гб места на жестком диске (SSD);
- операционная система Ubuntu 16.04, CentOS 7 или Debian 8 GNU/Linux.

Данная задача также рассматривается с точки зрения системного и вариантного анализа.

Системный анализ включает в себя [2]:

- системотехническое представление системы безопасности в виде «черного ящика»;

– описание входных и выходных данных;
– список функций, которые выполняет система безопасности;
– учет случайностей;
– декомпозицию системы и описание связей между ее элементами.

Вариантный анализ произведен исходя из выбранных критериев [3]:

– цена;
– масштабируемость;
– отказоустойчивость;
– интерфейсы управления.

# 2 ОБЗОР ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

Исторически, ситуация сложилась так, что слово «облако» используется в качестве метафоры сети Интернет. Позже, оно было использовано для изображения Интернет в компьютерных сетевых диаграммах и схемах.

## 2.1 Становление и развитие облачных вычислений

Облачные вычисления можно обозначить, как выделение ресурсов по требованию. В соответствии с Национальным институтом стандартов и технологий (NIST), формальное определение облачных вычислений заключается в следующем: «Облачные вычисления являются моделью обеспечения повсеместного, удобного доступа по требованию по сети, общему пулу конфигурируемых вычислительных ресурсов (например сетей, серверов, систем хранения данных (СХД), приложений и услуг), которые могут быстро и с минимальными усилиями предоставлены для управления поставщиком услуг» [5].

Согласно опросам института Понемон в 2016 г., среди 3476 респондентов в сфере информационной безопасности из Соединенных Штатов Америки, Великобритании, Австралии, Германии, Японии, Франции, России, Индии и Бразилии, 73% респондентов так или иначе используют облачные вычисления в своей инфраструктуре. Особый рост внедрения облачных услуг произошел в последние 2 года [6].

Хранение данных пользователей, почты и потребительских данных в облаке выросло в 2016 г. по сравнению с 2014 г. (рис. 2.1).

Облачные вычисления являются результатом объединения большого количества технологий и связующего ПО для обеспечения ресурсов, необходимых для решения задачи, балансировки процессов, мониторинга, автоматизации и прочего.

Основными отличиями предоставления облачных услуг от «классических» является использование:

– виртуализации;

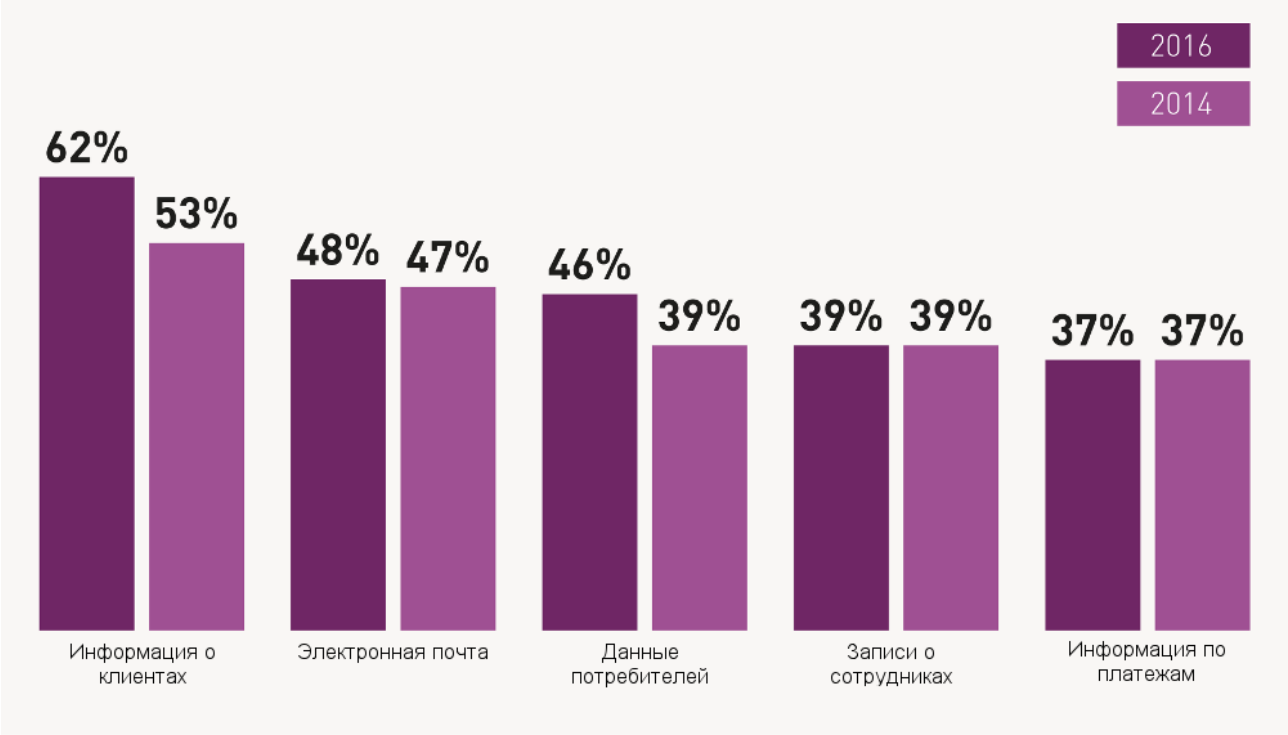


Рисунок 2.1 – Использование облака для хранения данных

– оркестратора;
– списка (каталога) услуг;
– портала самообслуживания;
– системы тарификации и выставления счетов (биллинга).

В вычислениях, виртуализация является процессом создания виртуальной версии чего-либо, в том числе аппаратных платформ виртуального компьютера, операционных систем (ОС), устройств хранения данных и вычислительных ресурсов.

Виртуализация может быть предоставлена на различных аппаратных и программных уровнях, таких как центральный процессор, диск, память, файловые системы и прочее. Чаще всего виртуализация используется для создания виртуальных машин и эмуляции различного оборудования для последующей установки операционных систем на них.

Виртуальные машины создаются на основе гипервизора, который работает поверх операционной системы хост-компьютера (физического компьютера, не виртуального). С помощью гипервизора возможна эмуляция аппаратных средств, таких как процессор, диск, сеть, память, а также установка гостевых

операционных систем на них. Возможно создание нескольких гостевых виртуальных машин с различными операционными системами на гипервизоре. Например, можно взять машину на Linux и установить ее на «голое» железо (bare-metal), и после настройки гипервизора возможно создание нескольких гостевых машин на Linux и Windows.

На данный момент все современные процессоры поддерживают аппаратную виртуализацию, это необходимо для безопасного и эффективного обмена ресурсами между хост-системой и гостевыми системами. Большинство современных процессоров и гипервизоров также поддерживают вложенную виртуализацию, что позволяет создавать виртуальные машины друг внутри друга.

Оркестратор является механизмом, выполняющим набор заданных операций по шаблону. В сервис-ориентированной архитектуре (SOA), оркестровка сервисов реализуется согласно стандарту BPEL. Это позволяет автоматизировать процессы создания услуг пользователей в облачной среде.

Список услуг предоставляется пользователю в виде шаблонов готовых тарифов на портале самообслуживания, однако существуют и «конфигураторы», которые позволяют пользователю создать шаблон индивидуально.

Портал самообслуживания является инструментом, с которым работает непосредственно пользователь. Именно на портале обслуживания размещается список услуг, доступных пользователю.

Система тарификации и выставления счетов является необходимым механизмом для определения финансовых затрат пользователя в соответствии с затраченными ресурсами пользователя.

## 2.2 Классификация облачных услуг

Поставщики облачных услуг (провайдеры) предлагают различные виды услуг, построенные поверх базового резервирования и освобождения ресурсов. Большинство из этих услуг попадают в одну из следующих категорий (рис. 2.2):

– инфраструктура как услуга (IaaS);

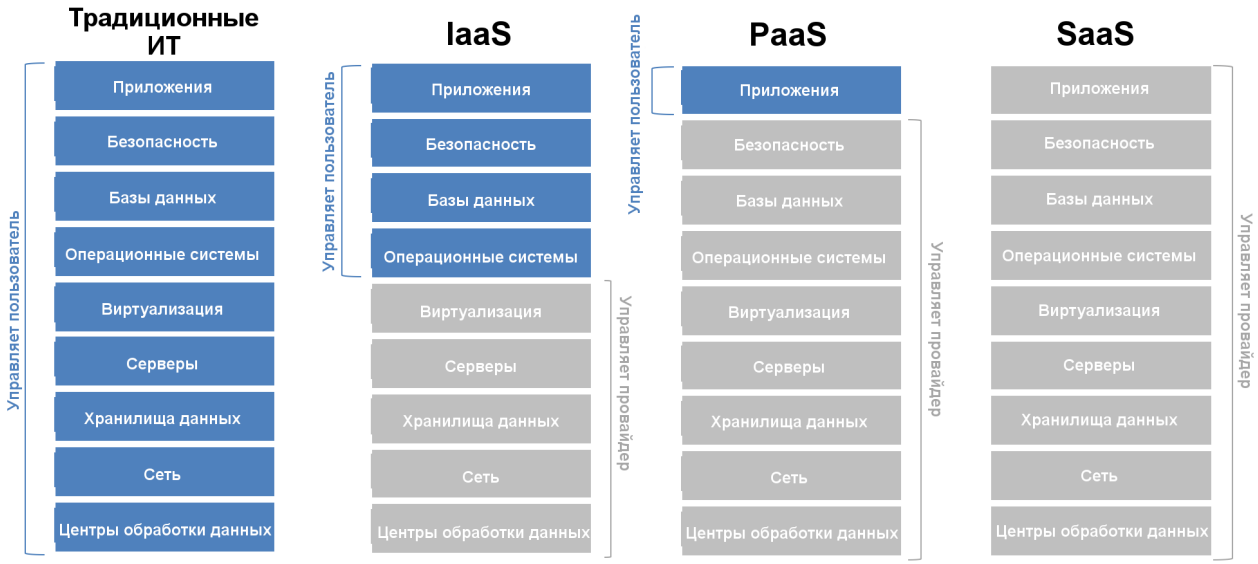


Рисунок 2.2 – Модели обслуживания облака

– платформа как услуга (PaaS);
– программное обеспечение как услуга (SaaS).

Большинство поставщиков облачных услуг используют различные виды программных интерфейсов, в том числе и веб-интерфейсов, на основе которых можно построить необходимый стек технологий. Облачные провайдеры используют модель «pay-as-you-go», в которой оплата производится только за время использования ресурсов.

Ключевыми функциями облачных вычислений являются:

– высокая скорость работы и гибкая масштабируемость;
– низкая стоимость услуг;
– легкий доступ к ресурсам;
– отсутствие необходимости в обслуживании оборудования;
– возможность совместного использования;
– надежность.

Доступ к необходимым ресурсам можно получить одним щелчком мыши, что экономит время и обеспечивает гибкость. В зависимости от потребностей услуги, можно легко масштабировать ресурсы как горизонтально, так и вертикально. Снижение первоначальных затрат на развертывание инфраструктуры позволяет сосредоточиться на приложениях и бизнесе. Компания имеет воз-

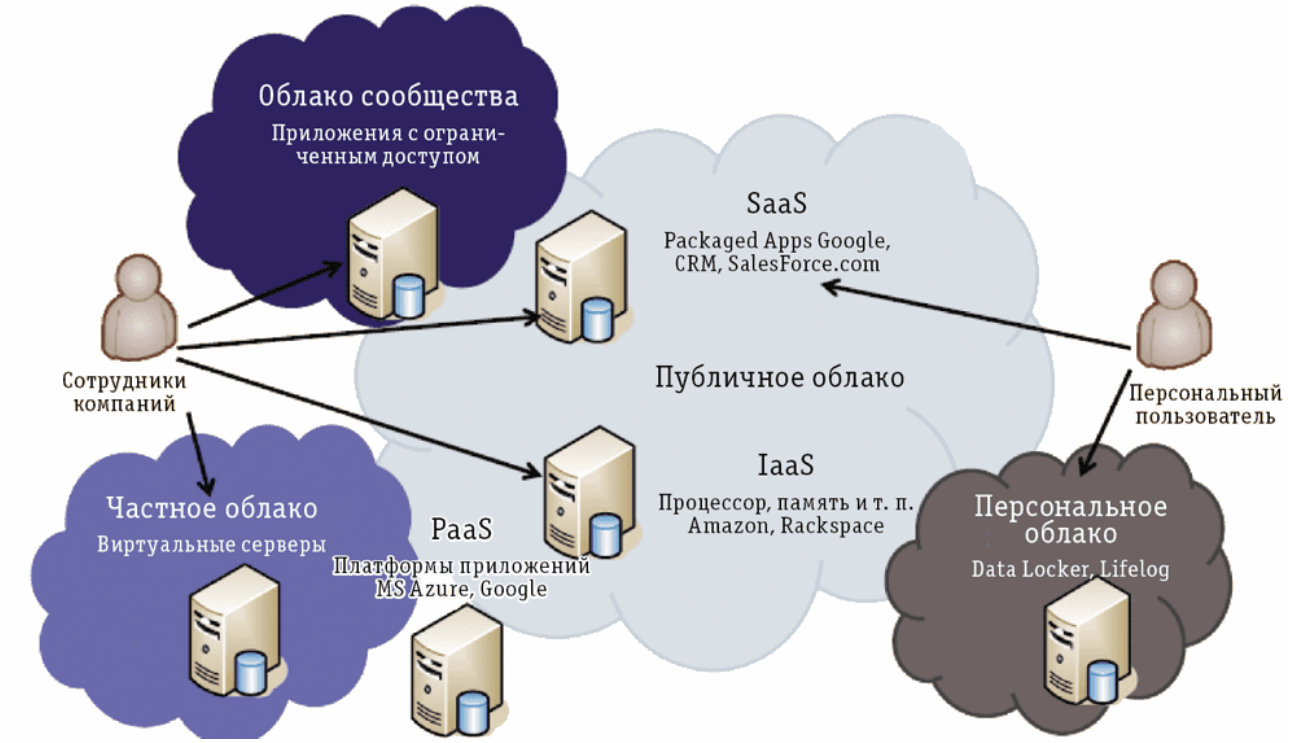


Рисунок 2.3 – Модели развертывания облака

можность заранее оценить стоимость, что значительно облегчает планирование бюджета. Пользователи могут получить доступ к инфраструктуре из любого места и устройства, до тех пор, пока существует интернет-подключение к поставщику. Все работы по техническому обслуживанию ресурсов осуществляются поставщиком облачных услуг, пользователи не должны беспокоиться об этом. Несколько пользователей могут использовать один и тот же пул доступных ресурсов. Ресурсы могут быть размещены в разных дата-центрах, для обеспечения повышенной надежности.

Широкое применение облачных вычислений позволяет использовать различные сценарии его использования. Как правило, облако может быть развернуто согласно следующим моделям (рис. 2.3):

– частное облако;
– публичное облако;
– общественное облако;
– гибридное облако.

Частное облако эксплуатируется исключительно одной организацией, оно может быть размещено внутри или снаружи сети организации и управ-

ляться внутренними командами или третьей стороной. Частное облако можно построить с использованием такого программного обеспечения, как OpenStack.

Публичное облако доступно для всех пользователей, любой может использовать его после предоставления платежных данных. Amazon Web Services (AWS) и Google Compute Engine (GCE) являются примерами публичных облаков.

Гибридное облако является результатом объединения публичных и частных облаков. Гибридное облако может быть использовано для хранения секретной информации о частном облаке, предлагая при этом услуги на основе этой информации из публичного.

Общественное облако, как правило, предназначено для сообщества или организации.

## 2.3 Стандартизация в области облачных вычислений

Облачные технологии относительно недавно вышли на рынок массового потребления, поэтому пока еще не существует четких общепринятых стандартов в сфере предоставления облачных услуг и обеспечении безопасности. Поставщики облачных услуг обходят эту проблему тремя путями.

Во-первых, облачные провайдеры создают собственные корпоративные стандарты, которые чаще всего публично не оглашаются. В таком случае потребитель может полагаться исключительно на репутацию компании, предоставляющей облачные услуги. Среди таких компаний можно выделить Google, Amazon, Microsoft, IBM, VMware, Oracle и прочие. Однако иногда такие как компании как IBM, участвуют в открытии облачных стандартов.

Во-вторых, компании адаптируют свои услуги согласно существующим и устоявшимся стандартам безопасности, проходят соответствующие сертификации с последующим получением свидетельства. Получение подобных сертификатов актуально в плане получения государственных и общественных заказов в долгосрочной перспективе [7].

В-третьих, различные правительственные, коммерческие и общественные организации принимают всяческие усилия по выработке требований к созданию безопасных облачных служб обработки информации.

Рабочая группа Object Management Group (OMG) в 2009 г. была инициатором создания Cloud Standarts Summit. Целью создания встречи является развитие информационных технологий (ИТ) и согласование стандартов по проблемам государственных облачных сред. В результате были созданы следующие рабочие группы:

- Cloud Security Alliance (CSA);
- Distributed Management Task Force (DMTF);
- Storage Networking Industry Association (SNIA);
- Open Grid Forum (OGF);
- Open Cloud Consortium (OCC);
- Organization for the Advancement of Structured Information Standards (OASIS);
- TM Forum;
- Internet Engineering Task Force (IETF);
- International Telecommunications Union (ITU);
- European Telecommunications Standards Institute (ETSI);
- National Institute of Standards and Technology (NIST);
- Object Management Group (OMG).

Наиболее известны достижения организаций NIST, CSA, OASIS, Open Data Center Alliance.

Cloud Security Alliance является некоммерческой организацией, созданной с целью продвижения идей обеспечения безопасности облачных вычислений, а также для повышения уровня осведомленности по данной тематике как поставщиков облачных услуг, так и потребителей. Ряд основных задач, выделяемых организацией CSA:

- поддержка взаимоотношений потребителей и поставщиков услуг в требованиях безопасности и контроля качества;
- независимые исследования по части защиты;

– разработка и внедрение программ повышения осведомленности и обеспечению безопасности;
– разработка руководств и методических рекомендаций по обеспечению безопасности.

Руководство по безопасности критических областей в области облачных вычислений (Security Guidance for Critical Areas of Focus in Cloud Computing) покрывает основные аспекты и дает рекомендации потребителям облачных сред в тринадцати стратегически важных областях:

– архитектурные решения сред облачных вычислений;
– государственное и корпоративное управление рисками;
– легальное и электронное открытие;
– соответствие техническим условиям и отчетность;
– управление жизненным циклом информации;
– портативность и совместимость;
– традиционная безопасность, непрерывность деятельности и восстановление в аварийных ситуациях;
– работа центра обработки данных;
– реакция на риски, уведомление и коррекционное обучение;
– прикладная безопасность;
– криптография и управление ключами;
– идентификация и управление доступом;
– виртуализация.

OASIS стимулирует развитие, сведение и принятие открытых стандартов для глобального информационного общества. Являясь источником многих современных основополагающих стандартов, организация видит облачные вычисления как естественное расширение сервис-ориентированной архитектуры и моделей управления сетью [8]. Технические агенты OASIS — это набор участников, многие из которых активно участвуют в построении моделей облаков, профилей и расширений на существующие стандарты. Примерами стандартов, разработанных в области политик безопасности, доступа и иден-

тификации, являются OASIS SAML, XACML, SPML, WS-SecurityPolicy, WS-Trust, WS-Federation, KMIP и ORMS.

Организация Open Data Center Alliance объявила о публикации двух моделей использования, призванных снять наиболее значимые препятствия на пути внедрения облачных вычислений. Первая модель использования называется «The Provider Security Assurance». В ней описаны требования к гранулированному описанию элементов обеспечения безопасности, которые должны предоставить поставщики услуг.

Вторая модель использования «The Security Monitoring» описывает требования к элементам, которые обеспечивают возможность мониторинга безопасности облачных услуг в реальном времени. В совокупности две модели использования формируют набор требований, который может стать основой для создания стандартной модели обеспечения безопасности облачных услуг и осуществления мониторинга этих услуг в реальном времени.

Национальный институт стандартов и технологий вместе с Американским национальным институтом стандартов (ANSI) участвует в разработке стандартов и спецификаций к программным решениям, используемым как в государственном секторе США, так и имеющим коммерческое применение. Сотрудники NIST разрабатывают руководства, направленные на описание облачной архитектуры, безопасность и стратегии использования, в числе которых руководство по системам обнаружения и предотвращения вторжений, руководство по безопасности и защите персональных данных при использовании публичных систем облачных вычислений.

В руководстве по системам обнаружения и предотвращения вторжений даются характеристики технологий IDPS и рекомендации по их проектированию, внедрению, настройке, обслуживанию, мониторингу и поддержке. Виды технологий IDPS различаются в основном по типам событий, за которыми проводится наблюдение, и по способам их применения. Рассмотрены следующие четыре типа IDPS-технологий: сетевые, беспроводные, анализирующие поведение сети и централизованные.

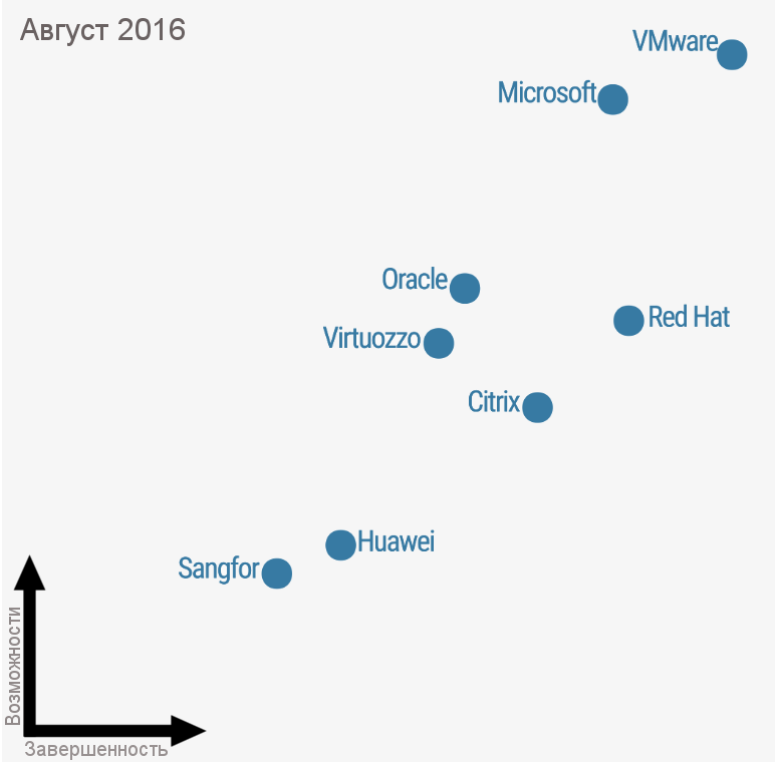


Рисунок 2.4 – «Магический квадрант Gartner» производителей систем виртуализации

В руководстве по безопасности и защите персональных данных при использовании публичных систем облачных вычислений в том числе дается обзор проблем безопасности и конфиденциальности, имеющих отношение к среде облачных вычислений: обнаружение атак на гипервизор, цели атак, отдельно рассматриваются распределенные сетевые атаки.

Инфраструктура как услуга является одной из форм облачных вычислений, которая обеспечивает доступ по требованию к физическим и виртуальным вычислительным ресурсам, сетям, межсетевым экранам, балансировщикам нагрузки и так далее. Для организации виртуальных вычислительных ресурсов, IaaS использует различные формы гипервизоров, таких как Xen, KVM, VMware ESX/ESXi, Hyper-V и прочие (рис. 2.4).

## 2.4 Обзор зарубежных облачных провайдеров

Amazon Web Services является одним из лидеров в области предоставления услуг различных облачных сервисов (рис. 2.5).

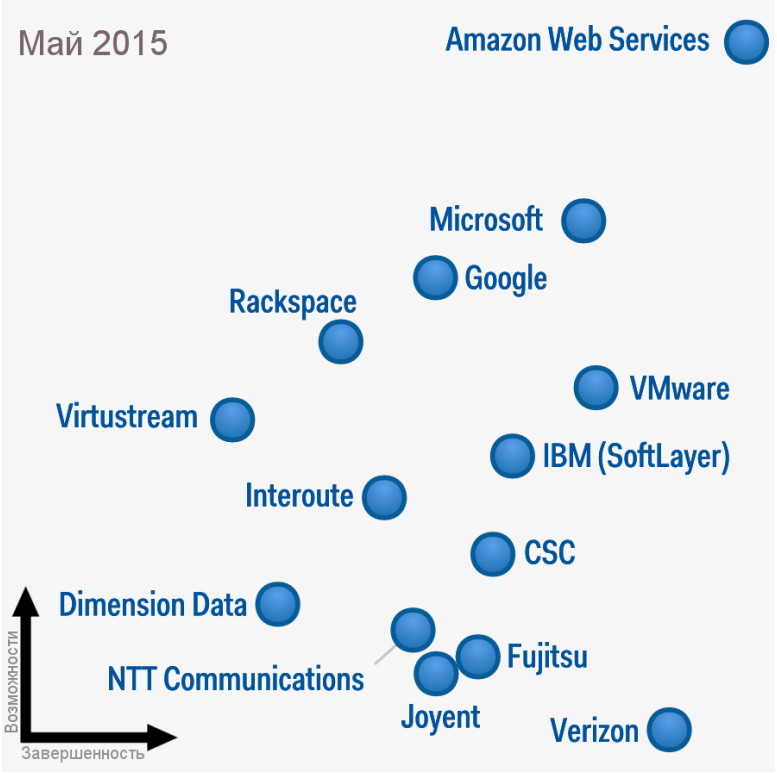


Рисунок 2.5 – «Магический квадрант Gartner» облачных провайдеров

С помощью Amazon Elastic Compute Cloud (EC2), Amazon предоставляет пользователям IaaS-инфраструктуру. Пользователь может управлять вычислительными ресурсами (инстансами) через веб-интерфейс Amazon EC2. Существует возможность горизонтального и вертикального масштабирования ресурсов, в зависимости от требований. AWS также предоставляет возможность управления инстансами посредством интерфейса командной строки и с помощью Application Programming Interface (API).

В качестве гипервизора, Amazon EC2 использует Xen [9]. Сервис предлагает инстансы различных конфигураций, которые можно выбрать в зависимости от требований. Некоторые примеры конфигураций виртуальных машин:

– t2.nano: 512 Мб ОЗУ, 1 vCPU (виртуальных процессорных ядер), 32 или 64-битные платформы;
– c4.large: 4 Гб ОЗУ, 2 vCPU, 64-битная платформа;
– d2.8xlarge: 256 Гб ОЗУ, 36 vCPU, 64-битная платформа, 10G Ethernet.

Amazon EC2 предоставляет некоторые предварительно настроенные образы операционных систем, называемые Amazon Machine Images (AMI). Эти

образы могут быть использованы для быстрого запуска виртуальных машин. Пользователь также может создавать собственные образы ОС. Amazon поддерживает настройки безопасности и доступа к сети для пользовательских виртуальных машин. С помощью Amazon Elastic Block Store (EBS) пользователь может монтировать хранилища данных к инстансам.

Amazon EC2 имеет много других возможностей, что позволяет:

– создавать «гибкие» IP-адреса для автоматического переназначения статического IP-адреса;
– предоставлять виртуальные частные облака;
– использовать услуги для мониторинга ресурсов и приложений;
– использовать автомасштабирование для динамического изменения доступных ресурсов.

Облачная платформа Azure, поддерживаемая компанией Microsoft, предлагает большой спектр облачных услуг, таких как: вычислительные мощности, платформы для мобильной и веб-разработки, хранилища данных, интернет вещей (IoT) и другие.

DigitalOcean позиционирует себя как простой облачный хостинг. Все виртуальные машины (дроплеты) работают под управлением гипервизора KVM и используют SSD-накопители. DigitalOcean предоставляет и другие функции, такие как IP-адреса расположенные в пределах одного дата-центра, частные сети, командные учетные записи и прочее. Простота веб-интерфейса, высокое качество работы виртуальных машин и доступность для обычного пользователя способствовали быстрому росту компании.

## 2.5 Развитие российского рынка облачных услуг

На российском рынке облачного хостинга все еще наблюдается большой рост. В связи с тем, что нет явно выраженного монополиста, таких как Amazon или Rackspace, российский рынок очень разнообразный. Конкуренция на рынке способствует значительному повышению качества предоставляемых услуг, а также гибкие тарифные планы и широкий перечень дополнительных услуг.

Также важную роль играет принятие Федерального закона от 21 июля 2014 г. № 242-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации в части уточнения порядка обработки персональных данных в информационно-телекоммуникационных сетях» [10]. Крупные компании обязаны хранить персональные данные пользователей на территории России, что способствует консолидации российского рынка облачных услуг.

Крупнейшие поставщики услуг ЦОД 2016 г. [11] представлены в табл. 2.1.

Таблица 2.1 – Крупнейшие поставщики услуг ЦОД в 2016 г.

| # | Название компании | Количество доступных стойко-мест | Количество размещенных стойко-мест | Загруженность мощностей (%) |
|---|---|---|---|---|
| 1 | Ростелеком | 3 900 | 3 432 | 88 |
| 2 | DataLine | 3 703 | 2 988 | 81 |
| 3 | DataPro | 3 000 | н/д | н/д |
| 4 | Linxtelecom | 2 040 | н/д | н/д |
| 5 | Selectel | 1 500 | 1 200 | 80 |
| 6 | Stack Group | 1 400 | 854 | 61 |
| 7 | Ай-Теко | 1 200 | 960 | 80 |
| 8 | DataSpace | 1 152 | 820 | 71 |
| 9 | SDN | 1 074 | 815 | 76 |
| 10 | Крок | 1 000 | 980 | 98 |

Данные DataLine включают показатели 7 ЦОД, расположенных на площадках OST и NORD. Данные по количеству введенных в эксплуатацию и реально размещенных стоек в ЦОД DataPro и Linxtelecom отсутствуют.

По итогам 2015 г. CNews Analytics впервые составил рейтинг крупнейших поставщиков IaaS. В исследовании приняли участие 14 компаний, совокупная выручка которых составила 3,8 млрд. рублей. По сравнению с 2014 г. участники заработали на 63% больше. Все участники рейтинга продемонстри-

ровали положительную динамику за исключением компании Inoventica (-3%). Высокие темпы роста свидетельствуют о том, что рынок IaaS находится в начале своего становления. Многие участники рейтинга вышли на этот рынок только в 2014-2015 г., чем объяснятся наличие большого числа компаний с ростом более в чем 3 раза: StackGroup (+733%), 1cloud.ru (+911%), CaravanAero (+1220%).

Сравнение крупнейших поставщиков IaaS в 2016 г. [11] представлено в табл. 2.2.

Таблица 2.2 – Крупнейшие поставщики IaaS в 2016 г.

| # | Название компании | Выручка IaaS в 2015 г. (тыс.р.) | Выручка IaaS в 2014 г. (тыс.р.) | ЦОД |
|---|---|---|---|---|
| 1 | ИТ-Град | 857 245 | 358 680 | Datalahti, DataSpace, SDN, AHOST |
| 2 | Крок | 667 609 | 440 315 | Волочаевская-1/2, Компрессор |
| 3 | Ай-Теко | 618 500 | 565 800 | ТрастИнфо |
| 4 | DataLine | 500 960 | 358 400 | NORD1/2/3/4, OST1/2/3 |
| 5 | SoftLine | 367 000 | 152 000 | н/д |
| 6 | Cloud4Y | 304 600 | 267 400 | Цветочная, M8/9/10, Nord, Equinix FR5, EvoSwitch |
| 7 | Stack Group | 182 900 | 21 948 | M1 |
| 8 | ActiveCloud | 103 702 | 56 910 | DataLine |
| 9 | Inoventica | 79 000 | 81 000 | н/д |
| 10 | 1cloud.ru | 78 307 | 7 743 | SDN, DataSpace |

Почти все участники рейтинга SaaS продемонстрировали положительную динамику выручки, при этом у 10 компаний оборот вырос более чем на

50%, а четыре поставщика облачных услуг зафиксировали рост выручки более чем на 100%: Naumen (+358%), amoCRM (+159%), ИТ-Град (+134%) и Artsofte (+125%).

Сравнение крупнейших поставщиков SaaS в 2016 г. [11] представлено в табл. 2.3.

Таблица 2.3 – Крупнейшие поставщики SaaS в 2016 г.

| # | Название компании | Выручка SaaS в 2015 г. (тыс.р.) | Выручка SaaS в 2014 г. (тыс.р.) | Рост выручки 2015/2014 г. (%) |
|---|---|---|---|---|
| 1 | СКБ Контур | 6 970 000 | 5 500 000 | 27 |
| 2 | Манго Телеком | 1 808 000 | 1 350 000 | 34 |
| 3 | B2B-Center | 1 163 300 | 1 155 842 | 1 |
| 4 | Барс Груп | 1 074 000 | 910 000 | 18 |
| 5 | SoftLine | 1 034 000 | 636 000 | 63 |
| 6 | Корпус Консалтинг СНГ | 783 511 | 602 825 | 32 |
| 7 | Terrasoft | 657 654 | 476 561 | 38 |
| 8 | Телфин | 398 500 | 317 900 | 25 |
| 9 | МойСклад | 395 000 | 265 000 | 49 |
| 10 | ИТ-Град | 265 400 | 113 420 | 134 |

## 2.6 Угрозы облачной безопасности

Своевременное определение основных угроз безопасности является первым шагом к минимизации рисков в сфере облачных вычислений. Исследовательская группа Cloud Security Alliance, на конференции RSA в Сан-Франциско, выделила 12 основных угроз безопасности в сфере облачных вычислений за 2016 г. [12]:

- утечка данных;
- компрометация учетных записей и обход аутентификации;
- взлом интерфейсов и API;

– уязвимость используемых систем;
– кража учетных записей;
– инсайдеры-злоумышленники;
– целевые кибератаки;
– перманентная потеря данных;
– недостаточная осведомленность;
– злоупотребление облачными сервисами;
– DDoS-атаки;
– совместные технологии, общие риски.

Как и в случае с традиционными инфраструктурами, облака подвергаются тем же угрозам. Злоумышленники пытаются получить доступ к данным пользователей, которые хранятся в облаке. Утечка данных компании может нанести значительный ущерб бизнесу, вплоть до банкротства. Поставщики облачных услуг пытаются минимизировать риски утечки информации путем внедрения многофакторной аутентификации и стойкого шифрования. К примеру все действия пользователя на портале самообслуживания должны осуществляться по безопасному протоколу HTTPS, а аутентификация в личный кабинет должна сопровождаться подтверждением через мобильный телефон или электронную почту.

Однако даже при использовании безопасного протокола, пользователь может использовать слабые пароли или управлять ключами шифрования ненадлежащим образом, например хранить их в публичном доступе. Также компании могут сталкиваться с проблемами назначения прав пользователей, например при увольнении сотрудника или при переводе его на другую должность. В таком случае, помимо повышения компетентности руководства компании, CSA рекомендует использовать одноразовые пароли, токены, USB-ключи, смарт-карты и прочие механизмы многофакторной аутентификации. Использование таких механизмов значительно усложняет подбор паролей и компрометация инфраструктуры возможна только при наличии физического доступа, например к USB-ключу.

Использование пользовательского интерфейса и API значительно упрощает взаимодействие пользователя с услугой. Однако появляется и дополнительный вектор для атаки злоумышленниками, так как информация предоставленная через API является строго конфиденциальной. Особо важно в данном случае проработать механизмы контроля доступа и шифровать API. Для предотвращения таких взломов необходимо периодически запускать тесты на проникновение, проводить аудит безопасности, моделировать угрозы и использовать прочие превентивные методы.

Наиболее популярная проблема при использовании облачных вычислений — это уязвимости в используемых приложениях. Зачастую, при использовании облачных IaaS-услуг, компании полностью полагаются на поставщика облачных услуг, хотя ответственность за размещаемые приложения и уязвимости в используемых системах лежит на пользователе. Для предотвращения взлома приложения, необходимо регулярное сканирование на предмет наличия уязвимостей, мониторинг активности приложения, применение последних патчей безопасности.

Путем кражи учетных записей пользователей облачного окружения, злоумышленники внедряют фишинговые страницы и различные эксплоиты. В таком случае стратегия «защита в глубину» может оказаться недостаточной. CSA рекомендует в таком случае выполнять контроль учетных записей пользователей, вести журналы выполняемых транзакций, а также использовать механизмы многофакторной аутентификации.

Уволенные сотрудники могут преследовать различные цели, начиная от мести и заканчивая коммерческой выгодой от кражи данных, поэтому особо важно контролировать учетные записи всех сотрудников компании, ключи шифрования, доступы к внутренним сервисам. Инсайдерские угрозы могут принести значительный ущерб компании, в случае утечки данных конкурентам. Также важно проводить мониторинг, аудит и логирование действий учетных записей.

В настоящее время особую опасность представляют целевые кибератаки. При таком методе взлома с помощью соответствующих инструментов можно

добиться проникновения в облачную инфраструктуру, при этом обнаружить такое вторжение затруднительно. Использование современных решений обеспечения безопасности позволяют вовремя обнаруживать такие кибератаки. Необходим мониторинг подобных инцидентов и незамедлительная реакция на них, также стоит использовать профилактические меры в целях недопущения взлома.

Перманентная потеря данных не настолько страшна, если использовать резервное копирование. Как правило, злоумышленники не ставят себе основной целью удаление данных, так как их можно восстановить при достаточной квалификации системных администраторов. Использование ежедневного резервного копирования и периодическая проверка работоспособности этих данных позволяет избежать их полной потери. Необходимо шифровать данные резервных копий, использовать альтернативные площадки для их размещения, разделять данные пользователей и данные приложений.

Компании, переходящие в облака часто не осведомлены о том, как это работает и с чем им придется в дальнейшем столкнуться. Для решения данного вопроса необходимо понимать как работают облачные сервисы, какие риски берет на себя компания при заключении договора с поставщиком облачных услуг. Персонал компании должен иметь соответствующую квалификацию для управления облачной услугой.

Облачные услуги могут использоваться злоумышленниками для совершения злонамеренных действий, таки как атак на отказ (DDoS), рассылка спама, размещение фишинговых страниц и прочее. Поставщики облачных услуг должны вести мониторинг таких пользователей, анализировать трафик и своевременно реагировать на подобные инциденты.

Особо актуальной угрозой на сегодняшний день стали DDoS-атаки. Вычислительные мощности дешевеют, поэтому подобные атаки становятся все дешевле и сильнее для злоумышленника. При выборе поставщика облачных услуг стоит обратить внимание, каким образом у них организована защита от подобных атак, так как при реальной угрозе провайдер может заблокировать пользователя и отказаться от предоставления услуг.

Облачные технологии базируются на большом количестве разнообразного программного обеспечения и протоколов. Если в одном компоненте возникает уязвимость, то она напрямую влияет на всю инфраструктуру. CSA рекомендует использовать стратегии «безопасности в глубину», сегментации сети, использования многофакторной аутентификации, использования систем обнаружения вторжений.

## 2.7 Тенденции развития облачных вычислений

В настоящее время одними из основных тенденций развития в сфере ИТ, в облачных вычислениях в частности, являются «зеленые» центры обработки данных и программно-конфигурируемые сети или Software-defined Networking (SDN).

Все центры обработки данных сталкиваются с проблемой вывода тепловой энергии. Тепловая энергия являет побочным действием от эксплуатации плотно укомплектованных серверных стоек. Для охлаждения ЦОД необходимо большое количество холодильных установок, которые потребляют количество энергии сравнимое с некоторыми городами [13]. Крупные компании хотят строить центры обработки данных в северных странах, таких как Исландия и Финляндия, однако удаленность от крупных магистральных сетевых точек и недостаток квалифицированных кадров для обслуживания ЦОД замедляют развитие данной тенденции. Тепло, выделяемое дата-центром можно использовать для обогрева близлежащих жилых комплексов. Возможность заново использовать излишнее тепло от серверов предусматривается на этапе разработки нового ЦОД, помогая повысить энергетическую эффективность оборудования.

Одной из главных расходных статей в проектировании центров обработки данных таких крупных компаний как Google, Amazon, eBay, Microsoft, является электроэнергия. В 2008 г. компания McKinsey & Company провела аналитический обзор выбросов углекислого газа от выработки электроэнергии для нужд ЦОД [14]. Согласно исследованиям, суммарные показатели выброса углекислого газа ЦОД сравнимы с выбросам такой страны как Аргентина.

За счет использования энергии солнца и ветра уже работает большое количество коммерческих ЦОД. Первый в мире «зеленый» дата-центр, который работает исключительно на энергии ветра построен в США, в штате Иллинойс.

Компания Microsoft в рамках Project Natick создала прототип ЦОД под названием Leona Philpot. Прототип был погружен в США, недалеко от Тихого океана и успешно проработал на протяжении четырех месяцев. Большое количество теплообменников, которыми был оснащен Leona Philpot, передавали лишнее тепло от серверов в воду. При этом по оценкам экологов, это не влияет на глобальное повышение на температуру воды в мировом океане.

Бывший сотрудник компании Google Джон Данн предложил использовать уже не работающую электростанцию в качестве нового ЦОД. Атомная электростанция Vermont Yankee использовалась с 1972 г. по 2014 г. Так как АЭС находится близко к водным ресурсам и железной дороге, необходимо было лишь преобразовать здание и подвести к ЦОД сетевую инфраструктуру.

В России существует проект строительства ЦОД рядом с Калининской АЭС. Таким образом корпорация Росэнергоатом обеспечит бесперебойным надежным источником электроэнергии для 4800 серверных стоек.

Основными тенденциями развития корпоративных сетей и сетей центров обработки данных являются:

- рост объемов трафика, изменение структуры трафика;
- рост числа пользователей мобильных приложений и социальных сетей;
- высокопроизводительные кластеры для обработки большого количества данных;
- виртуализация для предоставления облачных услуг.

Программно-конфигурируемая сеть — форма виртуализации вычислительных ресурсов (сети), в которой уровень управления отделен от уровня передачи данных (рис. 2.6). Основным преимуществом такой сети является то, что большое количество сетей возможно программно объединить в одну и централизованно управлять ею.

Если рассмотреть современный маршрутизатор или коммутатор, то логически его можно разделить на три компонента:

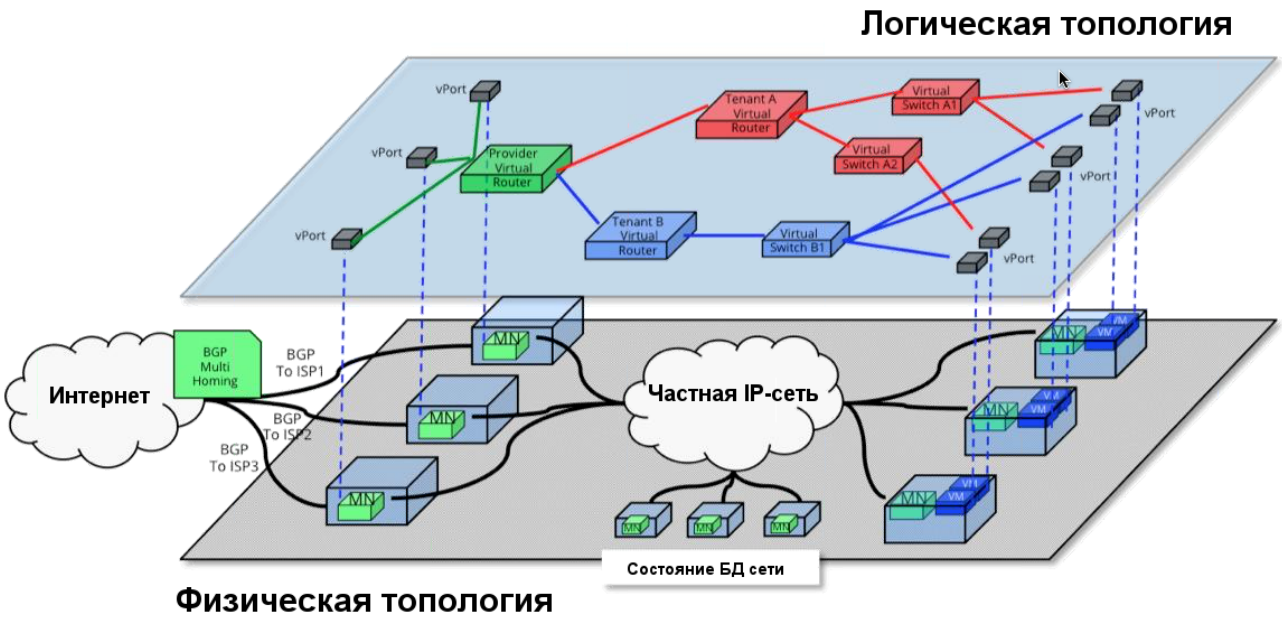


Рисунок 2.6 – Схема программно-конфигурируемой сети

– уровень управления;
– уровень управления трафиком;
– передача трафика.

На уровне управления присутствует командный интерфейс, программная оболочка, протоколы управления или встроенный веб-сервер. Основная задача уровня управления — обеспечить управление устройством. На уровень управления трафиком приходятся алгоритмы. Функциональная задача данного уровня — автоматическая реакция на изменение трафика. Функционал передачи трафика обеспечивает физическую передачу данных на низших уровнях.

Для построения программно-конфигурируемых сетей используется стандартный протокол OpenFlow. Большое количество коммутаторов уже поддерживает этот протокол. Крупные вендоры сетевого оборудования, такие как Hewlet-Packard, считают что развитие SDN и в частности протокола OpenFlow позволит возобновить процесс внедрения инноваций в уже давно устоявшейся области сетевых технологий.

## 2.8 Вопросы безопасности в соответствии с SPI

Безопасности инфраструктуры в соответствии с индексом параметра обеспечения безопасности (SPI) соответствуют безопасность уровня сети, уровня хоста и уровня приложений. Инфраструктура безопасности в большей степени применима для IaaS-клиентов, но также может быть применена для PaaS и SaaS, поскольку они имеют последствия для риска и соответствия руководству клиентов.

На сетевом уровне важно проводить различие между публичными и частными облаками. В частных облаках нет никаких новых атак, уязвимостей или изменений риска, которые должны быть приняты во внимание. С другой стороны, в публичных облаках, сети клиентов и облачного провайдера должны взаимодействовать между собой, что вносит изменения в требования безопасности.

Существует несколько факторов риска в этом:

- обеспечение конфиденциальности и целостности данных;
- обеспечение надлежащего контроля доступа (аутентификации, авторизации и аудита);
- обеспечение доступности интернет-ресурсов;
- замена установленной модели сетевых зон и уровней с доменами.

Данные, поступающие от поставщика публичного облака проходят через Интернет и это необходимо учитывать при обеспечении их конфиденциальности и целостности. Примером является Amazon Web Services, где в декабре 2008 г. использовался протокол HTTP вместо HTTPS, таким образом увеличивая риск изменения данных без ведома пользователя [15].

Отсутствие операций аудита сети облачного провайдера уменьшает доступ клиента к соответствующим журналам сети и данным. Этот фактор риска может привести к повторному использованию IP-адресов, которые находятся в кэше DNS и могут ввести в заблуждение пользователей.

Еще одним из факторов риска является BGP-перехват (префиксный перехват или перехват маршрута) является незаконным поглощением групп IP-адресов, повреждая интернет-таблицы маршрутизации. Из-за ошибки конфигурации, можно присвоить себе автономную систему (AS) без разрешения

владельца. Возможно и преднамеренное присвоение AS, но это случается гораздо реже.

Перехват префикса не является чем-то новым, однако из-за более широкого использования облачных вычислений, доступность облачных ресурсов увеличивается, а значит увеличивается количество мишеней для атак. Атаки на систему доменных имен (DNS) также не являются чем-то новым, однако и она представляет угрозу в сети из-за увеличения количества внешних DNS-запросов. В дополнение к уязвимости в протоколе DNS и его реализации, кэш DNS может подвергаться атакам. DNS-север может принимать некорректную информацию таким образом, что имя сервера целевого домена перенаправляется на другой целевой домен. DDoS также активно используется в атаках на IaaS. В таких протоколах как DNS, NTP, SNMP используется атака, при которой на маленький запрос к серверу генерируется огромный ответ, таким образом даже небольшим количеством ресурсов можно добиться, что весь сетевой канал будет занят паразитным трафиком. Кроме внешних атак, внутренний DDoS-атаки могут быть осуществлены через сеть поставщика IaaS-услуг. Скорее всего провайдер не контролирует подобное, поэтому предотвращение таких инцидентов может занять много времени. Только клиенты могут предотвратить атаки такого характера.

Традиционная сетевая безопасность зависит от изоляции модели сетевых зон и уровней, где пользователи и системы имеют доступ только к определенным зонам. Тем не менее, эта модель не может быть применена к публичным облакам IaaS и PaaS, но этот подход можно заменить в публичных облаках так называемыми «группами безопасности», «доменами безопасности» или «виртуальными центрами обработки данных». Эти группы безопасности могут позволить виртуальным машинам иметь доступ друг к другу с помощью виртуального межсетевого экрана, фильтрации трафика на основе IP-адреса, типов пакетов и портов. Кроме того, приложения логически сгруппированы на основе имен доменов.

Вопросы безопасности промежуточного узла (хоста) тесно связаны с различными моделями предоставления облачных услуг и моделей развертывания.

Хотя нет никаких новых угроз для хостов, которые являются специфическими для облачных вычислений, некоторые угрозы безопасности виртуализации (побег из виртуальной машины, слабый контроль над гипервизором, ошибки в конфигурации системы) сопровождаются последствиями. Эластичность облака может принести новые оперативные задачи с точки зрения обеспечения безопасности, которые могут быть гораздо сложнее обычных.

Так как злоумышленники могут использовать знания о конфигурации облака для вторжения в облачные сервисы, провайдеры облачных услуг публично не делятся информацией о используемых платформах, операционных системах и процессах. Таким образом, безопасность хоста не может быть детально известна пользователям и вся ответственность за обеспечение безопасности лежит на поставщике облачных услуг. Клиенты могут попросить провайдера обмениваться информацией в рамках соглашения о неразглашении (NDA), либо с помощью механизма оценки управления — ISO/IEC 27002 или SysTrust, чтобы дать клиенту гарантию. Для обеспечения виртуализации, поставщик облачных услуг использует гипервизоры Xen и VMware, которые являются bare-metal гипервизорами, то есть включены в состав существующей операционной системы. Таким образом уровень абстракции предоставляется пользователю, при этом скрывая операционную систему хоста от конечных пользователей. В случае SaaS, этот слой доступен только для разработчиков и персонала провайдера, а в случае PaaS, клиенты используют PaaS API, который взаимодействует с принимающим уровнем абстракции. Хотя поставщик облачных услуг несет ответственность за безопасность хоста для SaaS и PaaS услуг, клиент может управлять рисками хранения информации, размещенной в облаке.

При использовании IaaS, клиенты сами несут ответственность за безопасность, которая может быть отнесена к категории безопасности программного обеспечения и безопасности виртуального сервера.

Управление программным обеспечением для виртуализации относится к провайдеру, в то время как клиенты не имеют доступа к этому программному обеспечению, особенно если это публичное облако. Виртуализация аппаратного обеспечения или операционной системы позволяет совместно использовать

ресурсы на нескольких гостевых виртуальных машинах одновременно и не мешая друг другу. Виртуализации уровня хоста может быть осуществлено с использованием гипервизоров первого уровня, таких как VMware ESXi, Xen, Oracle VM VirtualBox, Hyper-V.

Клиенты имеют полный доступ к виртуальному экземпляру операционной системы, который виден из Интернета и изолирован от других экземпляров технологиями гипервизора. В таком случае безопасность и управление безопасностью лежит на пользователе. Публичные IaaS могут быть очень уязвимыми и стать жертвами новых направлений мошенничества, таких как кража закрытых SSH-ключей для доступа к хосту, кража счетов, атака на брандмауэры, развертывание троянов в виртуальных машинах. Для обеспечения высокого уровня безопасности виртуального сервер должны использоваться сильные операционные процедуры в сочетании с автоматизацией процедур:

- использование безопасной по умолчанию конфигурации — построение пользовательских образов виртуальных машин, которые имеют только возможности и услуги, необходимые для поддержки стека приложений;
- отслеживание обновлений гостевых операционных систем;
- защита целостности шаблонов от несанкционированного доступа;
- хранение закрытых ключей в безопасном месте;
- для доступа к командной оболочке не использовать пароли, а только SSH-ключи;
- периодически просматривать журналы для анализа подозрительных действий.

Безопасность приложений является одним из важнейших элементов безопасности инфраструктуры. Проектирование и внедрение приложений для облачной платформы требуют совершенствования существующей практики и стандартов для существующих программ безопасности приложений. Приложения варьируются от автономных однопользовательских приложений до сложных приложений электронной коммерции, многопользовательских, используемых миллионами пользователей, но наиболее уязвимыми являются веб-приложения. Так как браузер появился в качестве клиента конечного поль-

зователя для доступа к облачным приложениям, важно, чтобы безопасность браузера была включена в сферу безопасности приложений. Таким образом, пользователям рекомендуется регулярно проверять обновления браузера для поддержания безопасности.

Согласно некоторым исследованиям, почти половина уязвимостей относится к веб-приложениям [16]. Рейтинг OWASP Top 10 показывает, что основными угрозами для безопасности приложений являются SQL-инъекции, межсайтовый скриптинг (XSS), взломанные сессии, небезопасные ссылки, злонамеренное исполнение файлов и другие уязвимости, которые являются результатом ошибок программирования и конструктивных недостатков [17].

Веб-приложения, созданные и развернутые на платформе открытого облака легко сканировать, обладая соответствующими знаниями и инструментами злоумышленника, поэтому эти приложения должны быть разработаны в соответствии с моделью безопасности. Приложения должны включать в себя полный цикл разработки программного обеспечения, с проектированием, кодированием, тестированием и выпуском.

Ответственность за безопасность веб-приложений в облаке лежит как на поставщике облачных услуг, так и на пользователе и зависят от модели доставки облачных сервисов и соглашении об уровне обслуживания. В модели SaaS, поставщик услуг в основном отвечает за свою собственную безопасность приложений, в то время как клиент отвечает за оперативные функции безопасности и управления пользователями и доступом. Интересная проблема заключается в функции аутентификации и контроле доступа, предлагаемыми поставщиком SaaS. Различные поставщики предлагают разные методы: веб-инструменты администрирования пользовательского интерфейса для управления аутентификацией и управления доступом приложения (Salesforce, Google), встроенные функции, которые пользователи могут вызывать, назначать привилегии чтения и записи другим пользователям (Google Apps). Пользователи должны принять эти механизмы контроля доступа, а также включать в себя управление привилегиями на основе ролей и функций пользователя и реализовать политику надежного пароля.

Так как PaaS-облака поддерживают не только среду, но и собственные приложения пользователя, безопасность приложений можно разделить на два уровня: безопасность платформы PaaS и безопасность клиентских приложений, развернутых на PaaS. Поставщики облачных услуг несут ответственность за обеспечение работы стека программного обеспечения или гарантируют безопасность приложений сторонних разработчиков. Они также отвечают за мониторинг новых ошибок и уязвимостей и мониторинг совместно-используемой сетевой и системной инфраструктуры приложений. Разработчики PaaS должны разбираться в API конкретных облачных провайдеров и его функции безопасности: объекты безопасности и веб-сервисы для настройки аутентификации и авторизации инструментов в приложениях.

В IaaS, пользователи имеют полный контроль над своими приложениями, потому что весь стек ПО работает на виртуальных серверах клиента, и от провайдера облачных услуг они должны получать только основные рекомендации со ссылкой на политику брандмауэра. Веб-приложения, развернутые в публичном IaaS-облаке должны быть разработаны для модели Интернет, и должны периодически проверяться на наличие уязвимостей. Разработчики IaaS должны реализовать свои собственные функции для обработки аутентификации и авторизации и должны сделать их применение благоприятной для сервисных функций аутентификации.

# 3 СИСТЕМНЫЙ АНАЛИЗ

На данный момент при анализе и синтезе сложных программных и аппаратных систем все чаще используется системный подход. Важным моментом для системного подхода является определение структуры системы — совокупности связей между элементами системы, отражающих их взаимодействие.

Принципы системного анализа — это положения общего характера, являющиеся обобщением опыта работы человека со сложными системами. Пренебрежение принципами при проектировании любой нетривиальной технической системы, непременно приводит к потерям того или иного характера, от увеличения затрат в процессе проектирования до снижения качества и эффективности конечного продукта.

Системный анализ выполнен в соответствии с [2].

## 3.1 Принцип конечной цели

Принцип конечной цели — основополагающий принцип системного анализа. Конечная цель имеет абсолютный приоритет, в системе все должно быть подчинено достижению конечной цели. Принцип имеет несколько правил:

- для проведения системного анализа необходимо в первую очередь, сформулировать цель функционирования системы, так как расплывчатые, не полностью определенные цели влекут за собой неверные выводы;
- анализ системы следует вести на базе первоочередного уяснения основной цели исследуемой системы, что позволит определить ее существенные свойства, показатели качества и критерии оценки;
- при синтезе систем любая попытка изменения или совершенствования должна быть в первую очередь рассмотрена с позиции его полезности в достижении конечной цели;
- цель функционирования искусственной системы задается, как правило, системой, в которой исследуемая система является составной частью.

В соответствии с данным принципом должна быть четко сформулирована конечная цель — назначение проектируемой системы и сформирован список функций, которые должна выполнять система.

Цель проектирования — разработка системы безопасности облачной среды. Список функций проектируемой системы:

- Ф1 — авторизация и аутентификация пользователей;
- Ф2 — сетевая защита;
- Ф3 — идентификация и обработка инцидентов связанных с безопасностью;
- Ф4 — предоставление доступа к услугам;
- Ф5 — мониторинг.

## 3.2 Принцип единства

Принцип единства — это совместное рассмотрение системы как целого и как совокупности частей. Принцип ориентирован на декомпозицию с сохранением целостных представлений о системе. Система должна рассматриваться и как целое, и как совокупность элементов. Расчленение системы необходимо производить, сохраняя целостное представление о системе. Принцип подразумевает выделение подсистем, композиция которых в совокупности со связями позволяет выполнять все функции проектируемой системы, определяет ее структуру и может быть рассмотрена как единая, целостная система.

На основании функций проектируемой системы, представленных выше, в ней можно выделить следующие подсистемы:

а) подсистема аутентификации;
б) подсистема авторизации;
в) подсистема сетевой защиты;
г) подсистема проверки целостности данных.

На рис. 3.1 представлена схема взаимодействия между подсистемами.

Обозначения, приведенные на рис. 3.1 требуют пояснения:

а — информация, предоставляемая пользователем передается на сервер;

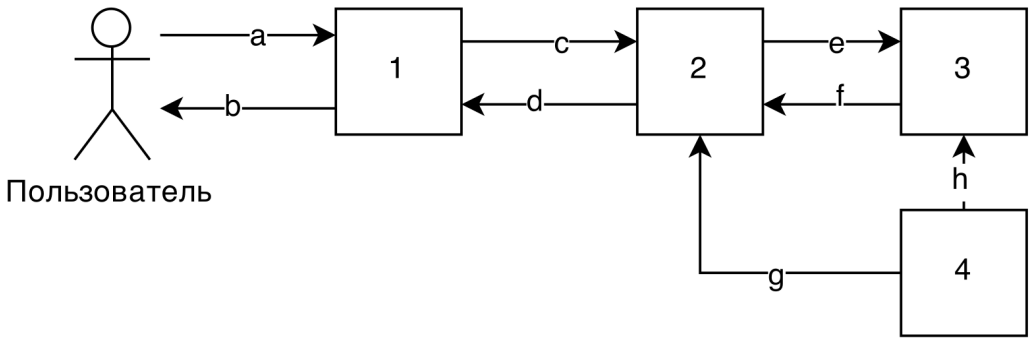


Рисунок 3.1 – Взаимодействие между подсистемами и их связь с окружающей средой

b — выходная информация (результат выполнения);

c — проверка корректности переданных данных;

d — в случае некорректных данных авторизации, возвращается управление к подсистеме аутентификации;

e — предоставление доступов к инфраструктуре;

f — возврат информации о правах доступа пользователя;

g — обеспечение целостности данных инфраструктуры;

h — обеспечение целостности данных пользователя.

## 3.3 Принцип связности и модульности

Любая часть системы должна рассматриваться со всеми своими связями с окружающими ее объектами, как внешними по отношению ко всей системе, так и внутренними — другими элементами системы. Если некоторая подсистема имеет связи только с внешней средой, то есть смысл реализовать ее в виде отдельной системы. Подсистема, не связанная ни с внешней средой, ни с другой подсистемой, является избыточной и должна быть удалена из системы.

В соответствии с принципом модульности, выделение модулей системы, если таковые имеются, продуктивно и оправдано. Под модулями здесь понимаются относительно автономные и достаточно простые блоки, выполняющие ограниченный набор функций. Модуль в отличие от подсистем, которые имеют нерегулярную структуру и, как правило, несут определенную функ-

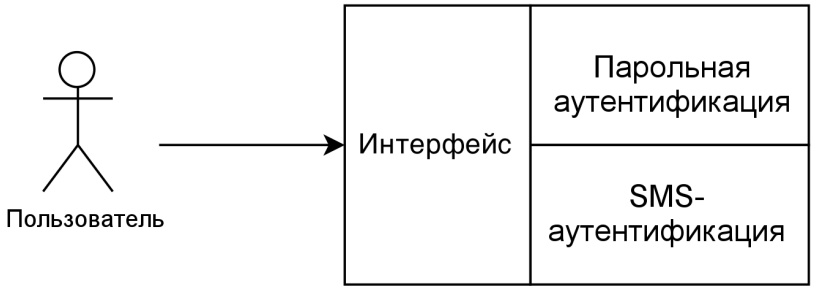


Рисунок 3.2 – Принцип модульности на примере подсистемы аутентификации

цию, частично отражающую функцию системы, имеет регулярную структуру и характерные для него внутренние и внешние связи. Принцип указывает на возможность вместо части системы исследовать совокупность ее входных и выходных воздействий. Выделению модулей соответствует декомпозиция сложной задачи на множество более простых подзадач.

Принцип модульности для разрабатываемой системы поясняется с помощью рис. 3.2, описывающего разбиение на модули системы аутентификации.

Излишняя детализация не требуется, поэтому остальные системы на модули принято решение не разбивать.

## 3.4 Принцип функциональности

Принцип определяет первичность функции по отношению к структуре, так же, как цель первична для функции. Другими словами, цель определяет функции системы, а функции определяют ее структуру — совокупность элементов с их связями. Структура же, в свою очередь, определяет параметры системы. В случае придания системе новых функций полезно пересматривать ее структуру, а не пытаться втиснуть новую функцию в старую схему. Поскольку выполняемые функции составляют процессы, то целесообразно рассматривать отдельно процессы, функции, структуры.

Функции подсистем приведены в пункте 3.1.

Матрица инциденций функций системы и функций назначения подсистем приведена в табл. 3.1.

Таблица 3.1 – Матрица инциденций

| Функции | Подсистемы | | | | Инфраструктура |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | |
| Ф1 | + | + | | | + |
| Ф2 | | | + | | + |
| Ф3 | | + | | | + |
| Ф4 | + | + | + | + | + |
| Ф5 | + | + | + | + | + |

В матрице инциденций знаком «+» обозначены функции, которые реализуются для каждой из подсистем.

Детализация функций подсистемы на примере подсистемы аутентификации:

а) обеспечение возможности ввода данных (веб-интерфейс или API);
б) обеспечение шифрованного канала для передачи данных;
в) сверка полученных данных с внутренней базой;
г) возможность восстановления доступов с помощью двухфакторной аутентификации.

Входными данными для подсистемы является информация о пользователе, а выходными — подтвержденный вход пользователя.

## 3.5 Принцип иерархии

Разработка иерархий классов является нетривиальной задачей. Грамотно спроектированные иерархии классов позволяют создавать высокоэффективные системы, плохо спроектированная иерархия приводит к созданию сложных и запутанных систем.

Выполнение принципа иерархичности для разрабатываемой системы на примере подсистемы проверки целостности данных проиллюстрировано на рис. 3.3.

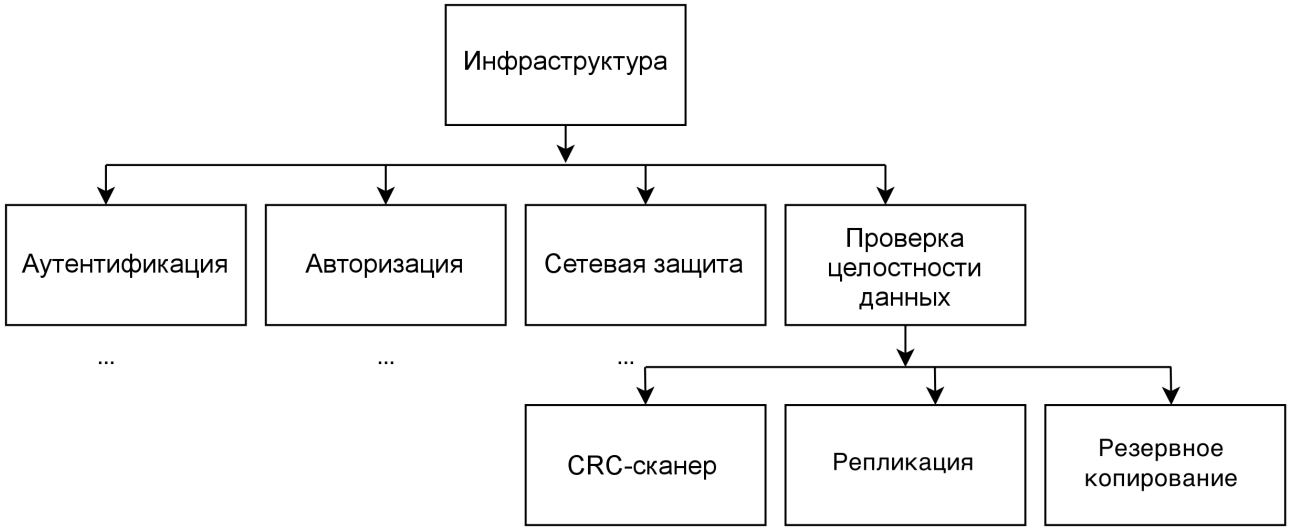


Рисунок 3.3 – Принцип иерархии на примере подсистемы обеспечения целостности данных

## 3.6 Принцип сочетания централизации и децентрализации

Степень централизации должна быть минимальной, обеспечивающей выполнение поставленной цели. Соотношение централизации и децентрализации определяется уровнями, на которых вырабатываются и принимаются управленческие решения.

Недостатком децентрализованного управления является увеличение времени адаптации системы, которое существенно влияет на функционирование системы в быстро меняющихся средах. То, что в централизованных системах можно сделать за короткий промежуток времени, в децентрализованной системе будет осуществляться весьма медленно. Недостатком централизованного управления является сложность управления из-за огромного потока информации, подлежащей переработке в старшей системе управления. В медленно меняющейся обстановке децентрализованная часть системы успешно справляется с адаптацией поведения системы к среде и с достижением глобальной цели системы за счет оперативного управления, а при резких изменениях среды осуществляется централизованное управление по переводу системы в новое состояние.

Например, можно выполнить декомпозицию подсистемы сетевой защиты таким образом:

а) подсистема работы виртуальной частной сети (VPN);

б) подсистема защиты от атак на отказ;

в) подсистема работы межсетевого экрана.

Такое разбиение позволит реализовать полученные подмножества в виде отдельных модулей.

### 3.7 Принцип развития

Учет изменяемости системы, ее способности к развитию, адаптации, расширению, замене частей, накапливанию информации. В основу синтезируемой системы требуется закладывать возможность развития, наращивания усовершенствования. Обычно расширение функций предусматривается за счет обеспечения возможности включения новых модулей, совместимых с уже имеющимися. С другой стороны, при анализе принцип развития ориентирует на необходимость учета предыстории развития системы и тенденций, имеющихся в настоящее время, для вскрытия закономерностей ее функционирования.

Одним из способов учета этого принципа разработчиками является рассмотрение системы относительно ее жизненного цикла. Условными фазами жизненного цикла являются: проектирование, изготовление, ввод в эксплуатацию, эксплуатация, наращивание возможностей (модернизация), вывод из эксплуатации (замена), уничтожение.

Отдельные авторы этот принцип называют принципом изменения (историчности) или открытости. Для того чтобы система функционировала, она должна изменяться, взаимодействовать со средой.

Проектируемая система может быть развита:

– переходом на более гибкие файловые системы;

– внедрением дополнительных технологий виртуализации;

– увеличением штата системных инженеров;

– введением расширенного мониторинга;

– многоуровневой защитой от атак на отказ;
– расширением памяти на системах хранения данных;
– многоуровневой репликацией и резервным копированием.

### 3.8 Принцип учета случайностей

Можно иметь дело с системой, в которой структура, функционирование или внешние воздействия не полностью определены.

Сложные системы не всегда подчиняются вероятностным законам. В таких системах можно оценивать «наихудшие» ситуации и проводить рассмотрение для них. Этот способ обычно называют методом гарантируемого результата, он применим, когда неопределенность не описывается аппаратом теории вероятностей.

События и действия, некорректные с точки зрения правил функционирования системы:

– попытка входа без использования двухфакторной аутентификации;
– сбой в работе аппаратуры, в частности устройств хранения данных;
– сбой сети в в пределах дата-центра или на магистрали;
– отсутствие своевременной работы системных инженеров;
– ошибки и уязвимости в используемых программных платформах.

Кроме того, необходимо вести контроль успешности и целостности проведения операций с компонентами системы, данными пользователей и корректно обрабатывать исключения, возникновение которых возможно в процессе работы системы.

Перечисленные выше принципы обладают высокой степенью общности. Такая интерпретация может привести к обоснованному выводу о незначимости какого-либо принципа. Однако, знание и учет принципов позволяет лучше увидеть существенные стороны решаемой проблемы, учесть весь комплекс взаимосвязей, обеспечить системную интеграцию.

# 4 ВАРИАНТНЫЙ АНАЛИЗ

Выберем гипервизор для организации виртуализации при помощи метода анализа иерархии (МАИ). Метод состоит в разложении проблемы на все более простые составные части и дальнейшей обработке последовательных суждений лица принимающего решение по парным сравнениям. В результате может быть выражена интенсивность или относительная степень взаимодействия элементов в иерархии. В результате получаются численные выражения этих суждений. МАИ включает в себя процедуры синтеза множественных суждений, получение приоритетных критериев и нахождение альтернативных решений. Полученные знания являются оценками в шкале отношений и соответствуют жестким оценкам [3].

В качестве альтернатив используются различные гипервизоры, на которых может быть реализована виртуализация для облачной среды. В зависимости от выбранного гипервизора будет выбран тот или другой интерфейс взаимодействия с пользователем, так как каждая система имеет в своем составе такой интерфейс. Критерием эффективности являются: цена, масштабируемость, отказоустойчивость, интерфейсы управления.

Альтернативы, которые участвуют в вариантном анализе:

- KVM (альтернатива А);
- Hyper-V (альтернатива Б);
- VMware vSphere (альтернатива В).

Критерии, по которым выбирается тот, или иной гипервизор:

- цена (А1);
- масштабируемость (А2);
- отказоустойчивость (А3);
- интерфейсы управления (А4).

## 4.1 Построение матриц парных сравнений второго уровня

На основе вышеперечисленных критериев построим матрицу парных сравнений второго уровня, где строки и столбцы составляют выбранные критерии. Сравнение критериев проведём по шкале относительной важности согласно с табл. 4.1.

Таблица 4.1 – Оценка критериев

| Интенсивность относительной важности | Определение |
|---|---|
| 1 | если элементы $A_i$ и $A_k$ одинаково важны |
| 3 | если элементы $A_i$ и $A_k$ одинаково важны |
| 5 | если элемент $A_i$ значительно важнее элемента $A_k$ |
| 7 | если элемент $A_i$ явно важнее элемента $A_k$ |
| 9 | если элемент $A_i$ по своей значимости абсолютно превосходит элемент $A_k$ |
| 2,4,6,8 | используются для облегчения компромиссов между оценками, слегка отличающимися от основных чисел |

В результате выполнения попарных сравнений, построили матрицу, представленную в табл. 4.2.

Таблица 4.2 – Матрица попарных сравнений второго уровня

| Критерии | | A1 | A2 | A3 | A4 |
|---|---|---|---|---|---|
| A1 | Цена | 1 | 1/5 | 1/7 | 3 |
| A2 | Масштабируемость | 5 | 1 | 1/5 | 7 |
| A3 | Отказоустойчивость | 7 | 5 | 1 | 8 |
| A4 | Интерфейсы управления | 1/3 | 1/7 | 1/8 | 1 |

## 4.2 Вычисление вектора приоритетов для матрицы парных сравнений второго уровня

Из группы матриц попарных сравнений формируется набор локальных приоритетов, которые выражают относительное влияние множества элементов на элемент примыкающего сверху уровня. Сначала вычислим геометрическое среднее в каждой строке матрицы А по формуле (4.1):

$$b_i = \sqrt[n]{\prod_{k=1}^{n} a_{ik}} \tag{4.1}$$

Проведем вычисления компонент вектора локальных приоритетов:

$b_1 = \sqrt[4]{1 \cdot 0,2 \cdot 0,1429 \cdot 3} = \sqrt[4]{0,0857} = 0,5411$

$b_2 = \sqrt[4]{5 \cdot 1 \cdot 0,2 \cdot 7} = \sqrt[4]{7} = 1,6266$

$b_3 = \sqrt[4]{7 \cdot 5 \cdot 1 \cdot 8} = \sqrt[4]{280} = 4,0906$

$b_4 = \sqrt[4]{0,3333 \cdot 0,1429 \cdot 0,125 \cdot 1} = \sqrt[4]{0,006} = 0,2783$

Просуммируем полученные значения:

$B = 0,5411 + 1,6266 + 4,0906 + 0,2783 = 6,5355$

Определим значения компонент вектора локальных приоритетов по формуле (4.2):

$$x_i = \frac{b_i}{B}, i = \overline{1,n} \tag{4.2}$$

Выполним расчеты:

$x_1 = \frac{b_1}{B} = \frac{0,5411}{6,5355} = 0,0828$

$x_2 = \frac{b_2}{B} = \frac{1,6266}{6,5355} = 0,2489$

$x_3 = \frac{b_3}{B} = \frac{4,0906}{6,5355} = 0,6259$

$x_4 = \frac{b_4}{B} = \frac{0,2783}{6,5355} = 0,0426$

Так как числа $b_i$ нормализуются делением каждого числа на сумму всех чисел, то должно выполняться условие (4.3):

$$\sum_{i=1}^{n} x_i = 1, i = \overline{1,n} \tag{4.3}$$

В итоге:

$X = 0,0828 + 0,2489 + 0,6259 + 0,0426 = 1,0002$

Погрешность в $0,0002$ допустима и является следствием округления до четвертого знака.

4.3 Исследование на согласованность матрицы парных сравнений второго уровня

Оценим отношение согласованности для матрицы попарных сравнений второго уровня по формуле (4.4):

$$y_i = \sum_{i=1}^{n} a_{ik}, k = 1, 2, ..., n \quad (4.4)$$

Выполним расчеты:

$y_1 = 1 + 5 + 7 + 0,3333 = 13,3333$

$y_2 = 0,2 + 1 + 5 + 0,1429 = 6,3429$

$y_3 = 0,1429 + 0,2 + 1 + 0,125 = 1,4679$

$y_4 = 3 + 7 + 8 + 1 = 19$

Вычислим наибольшее собственное значение матрицы сравнений согласно формуле (4.5):

$$\lambda_{max} = \sum_{i=1}^{n} x_i \cdot y_i \quad (4.5)$$

где $x_i$ — значения компонент вектора локальных приоритетов.

$\lambda_{max} = 0,0828 \cdot 13,3333 + 0,2489 \cdot 6,3429 + 0,6259 \cdot 1,4679 + 0,0426 \cdot 19 = 1,104 + 1,5787 + 0,9188 + 0,8094 = 4,4109$

Положительная обратно-симметричная матрица является согласованной тогда и только тогда, когда порядок матрицы и ее наибольшее собственное значение совпадают ($\lambda_{max} = n$).

Если элементы положительной обратносимметричной согласованной матрицы А изменить незначительно, то максимальное собственное значение $\lambda_{max}$ также изменится незначительно. Если $\lambda_{max} \neq n$, всегда $\lambda_{max} > n$.

Как и ожидалось:

$\lambda_{max} = 4,4109 > n = 4$

В качестве степени отклонения положительной обратно-симметричной матрицы А от согласованной матрицы принимается отношение (4.6):

$$\text{ИС} = \frac{\lambda_{max} - n}{n - 1} \quad (4.6)$$

которое называется индексом согласованности (ИС) матрицы А и является показателем близости этой матрицы к согласованной.

Вычислим индекс согласованности для данной задачи:

$ИС = \frac{4{,}4109-4}{3} = 0,137$

Теперь необходимо сравнить значение индекса согласованности со значением случайной согласованности (СС).

Таблица 4.3 – Случайная согласованность

| Размер матрицы n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Случайная согласованность | 0 | 0 | 0,58 | 0,9 | 1,12 | 1,24 | 1,32 | 1,41 | 1,45 | 1,49 |

Если разделить индекс согласованности на число, соответствующее случайной согласованности матрицы того же порядка, получается отношение согласованности (ОС), формула (4.7):

$$ОС = \frac{ИС}{СС} \cdot 100\% \quad (4.7)$$

Величина отношения согласованности должна быть порядка 10% или менее, чтобы быть приемлемой. На практике же допускается значение, не превышающее 20% [4]. Если значение отношения согласованности выходит из этих пределов, то экспертам нужно исследовать задачу и пересмотреть суждения:

$ОС = \frac{0{,}137}{0{,}9} \cdot 100\% = 15,2\%$

Полученное значение $ОС = 15,2\% < 20\%$. Считаем, что матрица попарных сравнений второго уровня является согласованной.

Оценки предпочтений критериев лица принимающего решение (ЛПР) представлены в табл. 4.4.

Таблица 4.4 – Численные оценки предпочтений критериев ЛПР

| Критерии | | Место | Вес |
|---|---|---|---|
| A3 | Отказоустойчивость | 1 | 0,6259 |
| A2 | Масштабируемость | 2 | 0,2489 |
| A1 | Цена | 3 | 0,0828 |
| A4 | Интерфейсы управления | 4 | 0,0426 |

Исходя из вычисленных значений численных оценок предпочтения делаем вывод, что критерий «Отказоустойчивость» является наиболее важным. Критерий «Масштабирование» имеет существенный вес, а критерии «Цена» и «Интерфейсы управления» малозначимы.

## 4.4 Построение матриц попарных сравнений третьего уровня

На третьем уровне МАИ для каждого критерия проводятся попарные сравнения альтернатив и реализуются этап синтеза локальных приоритетов $z_j$ (j — номер альтернативы, $j = \overline{1, m}$ в нашем примере $m = 3$) в соответствии с формулами (4.1) ... (4.7). Также проводится исследование матрицы на согласованность [3].

### 4.4.1 Критерий «Цена»

В табл. 4.5 проведены попарные сравнения альтернатив по критерию A1 «Цена».

Таблица 4.5 – Матрица попарных сравнений по критерию A1

| Альтернатива | M1 | M2 | M3 |
|---|---|---|---|
| M1 | 1 | 4 | 6 |
| M2 | 1/4 | 1 | 3 |
| M3 | 1/6 | 1/3 | 1 |

Согласно формуле (4.1) вычислим сравнительную желательность альтернатив по первому критерию:

$b_1 = \sqrt[3]{1 \cdot 4 \cdot 6} = \sqrt[3]{24} = 2,8845$

$b_2 = \sqrt[3]{0,25 \cdot 1 \cdot 3} = \sqrt[3]{0,75} = 0,9086$

$b_3 = \sqrt[3]{0,1667 \cdot 0,3333 \cdot 1} = \sqrt[3]{0,0556} = 0,3817$

Просуммируем полученные значения:

$B = 2,8845 + 0,9086 + 0,3817 = 4,1748$

Далее воспользуемся формулой (4.2), заменив идентификаторы $x_i$ на $z_j$:

$z_1 = \frac{2,8845}{4,1748} = 0,6909$

$z_2 = \frac{0,9086}{4,1748} = 0,2176$

$z_3 = \frac{0,3817}{4,1748} = 0,0914$

Проведем проверку по формуле (4.3):

$\sum_{i=1}^{3} 0,6909 + 0,2176 + 0,0914 = 0,9999$

Оценка погрешности вычисляется по формуле (4.8):

$$\delta_x = \frac{|1 - \sum_{i=1}^{n} z_i|}{1} \cdot 100\%, i = \overline{1, n} \quad (4.8)$$

Оценим погрешность вычислений:

$\delta_x = \frac{|1-0,9999|}{1} \cdot 100\% = 0,01$

Оценим отношение согласованности для матрицы попарных сравнений второго уровня по формуле (4.4):

$y_1 = 1 + 0,25 + 0,1667 = 1,4167$

$y_2 = 4 + 1 + 0,3333 = 5,3333$

$y_3 = 6 + 3 + 1 = 10$

Вычислим наибольшее собственное значение матрицы сравнений согласно (4.5):

$\lambda_{max} = 0,6909 \cdot 1,4167 + 0,2176 \cdot 5,3333 + 0,0914 \cdot 10 = 0,9788 + 1,1605 + 0,914 = 3,0533$

Вычислим индекс согласованности для данной матрицы:

ИС $= \frac{3,0533-3}{2} = 0,0267$

Далее найдем отношение согласованности:

ОС $= \frac{0,0267}{0,58} \cdot 100\% = 4,6\%$

Полученное значение ОС $= 4,6\% < 20\%$. Считаем, что матрица парных сравнений третьего уровня по критерию А1 является согласованной.

### 4.4.2 Критерий «Масштабируемость»

В табл. 4.6 проведены попарные сравнения альтернатив по критерию А2 «Масштабируемость».

Таблица 4.6 – Матрица попарных сравнений по критерию А2

| Альтернатива | М1 | М2 | М3 |
|---|---|---|---|
| М1 | 1 | 3 | 1 |
| М2 | 1/3 | 1 | 1/3 |
| М3 | 1 | 3 | 1 |

Согласно формуле (4.1) вычислим сравнительную желательность альтернатив по первому критерию:

$b_1 = \sqrt[3]{1 \cdot 3 \cdot 1} = \sqrt[3]{3} = 1,4422$

$b_2 = \sqrt[3]{0,3333 \cdot 1 \cdot 0,3333} = \sqrt[3]{0,1111} = 0,4807$

$b_3 = \sqrt[3]{1 \cdot 3 \cdot 1} = \sqrt[3]{3} = 1,4422$

Просуммируем полученные значения:

$\boldsymbol{B} = 1,4422 + 0,4807 + 1,4422 = 3,3651$

Далее воспользуемся формулой (4.2), заменив идентификаторы $x_i$ на $z_j$:

$z_1 = \frac{1,4422}{3,3651} = 0,4286$

$z_2 = \frac{0,4807}{3,3651} = 0,1429$

$z_3 = \frac{1,4422}{3,3651} = 0,4286$

Проведем проверку по формуле (4.3):

$\sum_{i=1}^{3} 0,4286 + 0,1429 + 0,4286 = 1,0001$

Оценим погрешность вычислений:

$\delta_x = \frac{|1-1,0001|}{1} \cdot 100\% = 0,01$

Оценим отношение согласованности для матрицы попарных сравнений второго уровня по формуле (4.4):

$y_1 = 1 + 0,3333 + 1 = 2,3333$

$y_2 = 3 + 1 + 3 = 7$

$y_3 = 1 + 0,3333 + 1 = 2,3333$

Вычислим наибольшее собственное значение матрицы сравнений согласно (4.5):

$\lambda_{max} = 0,4286 \cdot 2,3333 + 0,1429 \cdot 7 + 0,4286 \cdot 2,3333 = 1,0001 + 1,0003 + 1,0001 = 3,0005$

Вычислим индекс согласованности для данной матрицы:

ИС $= \frac{3,0005-3}{2} = 0,0003$

Далее найдем отношение согласованности:

ОС $= \frac{0,0003}{0,58} \cdot 100\% = 0,05\%$

Полученное значение ОС $= 0,05\% < 20\%$. Считаем, что матрица парных сравнений третьего уровня по критерию А2 является согласованной.

### 4.4.3 Критерий «Отказоустойчивость»

В табл. 4.7 проведены попарные сравнения альтернатив по критерию А3 «Отказоустойчивость».

Таблица 4.7 – Матрица попарных сравнений по критерию А3

| Альтернатива | М1 | М2 | М3 |
|---|---|---|---|
| М1 | 1 | 1/3 | 1/3 |
| М2 | 3 | 1 | 1/3 |
| М3 | 3 | 3 | 1 |

Согласно формуле (4.1) вычислим сравнительную желательность альтернатив по первому критерию:

$b_1 = \sqrt[3]{1 \cdot 0,3333 \cdot 0,3333} = \sqrt[3]{0,1111} = 0,4807$

$b_2 = \sqrt[3]{3 \cdot 1 \cdot 0,3333} = \sqrt[3]{0,9999} = 1$

$b_3 = \sqrt[3]{3 \cdot 3 \cdot 1} = \sqrt[3]{9} = 2,0801$

Просуммируем полученные значения:

$\boldsymbol{B} = 0,4807 + 1 + 2,0801 = 3,5608$

Далее воспользуемся формулой (4.2), заменив идентификаторы $x_i$ на $z_j$:

$z_1 = \frac{0,4807}{3,5608} = 0,135$

$z_2 = \frac{1}{3,5608} = 0,2808$

$z_3 = \frac{2,0801}{3,5608} = 0,5842$

Проведем проверку по формуле (4.3):

$\sum_{i=1}^{3} 0,135 + 0,2808 + 0,5842 = 1$

Оценим погрешность вычислений:

$\delta_x = \frac{|1-1|}{1} \cdot 100\% = 0$

Оценим отношение согласованности для матрицы попарных сравнений второго уровня по формуле (4.4):

$y_1 = 1 + 3 + 3 = 7$

$y_2 = 0,3333 + 1 + 3 = 4,3333$

$y_3 = 0,3333 + 0,3333 + 1 = 1,6666$

Вычислим наибольшее собственное значение матрицы сравнений согласно (4.5):

$\lambda_{max} = 0,135 \cdot 7 + 0,2808 \cdot 4,3333 + 0,5842 \cdot 1,6666 = 0,945 + 1,2168 + 0,9736 = 3,1354$

Вычислим индекс согласованности для данной матрицы:

$\text{ИС} = \frac{3,1354-3}{2} = 0,0677$

Далее найдем отношение согласованности:

$\text{ОС} = \frac{0,0677}{0,58} \cdot 100\% = 11,67\%$

Полученное значение $\text{ОС} = 11,67\% < 20\%$. Считаем, что матрица парных сравнений третьего уровня по критерию А3 является согласованной.

### 4.4.4 Критерий «Интерфейсы управления»

В табл. 4.8 проведены попарные сравнения альтернатив по критерию А4 «Интерфейсы управления».

Таблица 4.8 – Матрица попарных сравнений по критерию А4

| Альтернатива | М1 | М2 | М3 |
|---|---|---|---|
| М1 | 1 | 1/3 | 1/3 |
| М2 | 3 | 1 | 1/3 |
| М3 | 3 | 3 | 1 |

Согласно формуле (4.1) вычислим сравнительную желательность альтернатив по первому критерию:

$b_1 = \sqrt[3]{1 \cdot 0,3333 \cdot 0,3333} = \sqrt[3]{0,1111} = 0,4807$

$b_2 = \sqrt[3]{3 \cdot 1 \cdot 0,3333} = \sqrt[3]{0,9999} = 1$

$b_3 = \sqrt[3]{3 \cdot 3 \cdot 1} = \sqrt[3]{9} = 2,0801$

Просуммируем полученные значения:

$B = 0,4807 + 1 + 2,0801 = 3,5608$

Далее воспользуемся формулой (4.2), заменив идентификаторы $x_i$ на $z_j$:

$z_1 = \frac{0,4807}{3,5608} = 0,135$

$z_2 = \frac{1}{3,5608} = 0,2808$

$z_3 = \frac{2,0801}{3,5608} = 0,5842$

Проведем проверку по формуле (4.3):

$\sum_{i=1}^{3} 0,135 + 0,2808 + 0,5842 = 1$

Оценим погрешность вычислений:

$\delta_x = \frac{|1-1|}{1} \cdot 100\% = 0$

Оценим отношение согласованности для матрицы попарных сравнений второго уровня по формуле (4.4):

$y_1 = 1 + 3 + 3 = 7$

$y_2 = 0,3333 + 1 + 3 = 4,3333$

$y_3 = 0,3333 + 0,3333 + 1 = 1,6666$

Вычислим наибольшее собственное значение матрицы сравнений согласно (4.5):

$\lambda_{max} = 0,135 \cdot 7 + 0,2808 \cdot 4,3333 + 0,5842 \cdot 1,6666 = 0,945 + 1,2168 + 0,9736 = 3,1354$

Вычислим индекс согласованности для данной матрицы:

$ИС = \frac{3,1354-3}{2} = 0,0677$

Далее найдем отношение согласованности:

$ОС = \frac{0,0677}{0,58} \cdot 100\% = 11,67\%$

Полученное значение $ОС = 11,67\% < 20\%$. Считаем, что матрица парных сравнений третьего уровня по критерию А4 является согласованной.

## 4.5 Анализ результатов оценки альтернатив

По полученным значениям векторов локальных приоритетов сделаем выводы о важности альтернатив для каждого из критериев.

Критерий «Цена»:

а) альтернатива А (0, 6909);

б) альтернатива Б (0, 2176);

в) альтернатива В (0, 0914).

Альтернатива А имеет явное преимущество над альтернативой Б, которая в свою очередь имеет явное преимущество над альтернативой В.

Критерий «Масштабируемость»:

а) альтернатива А (0, 4286);

б) альтернатива Б (0, 1429);

в) альтернатива В (0, 4286).

Альтернативы А и В имеют равные преимущества над альтернативой Б.

Критерий «Отказоустойчивость»:

а) альтернатива А (0, 135);

б) альтернатива Б (0, 2808);

в) альтернатива В (0, 5842).

Альтернатива В имеет преимущество над альтернативой Б, которая в свою очередь имеет преимущество перед альтернативой А.

Критерий «Интерфейсы управления»:

а) альтернатива А (0, 135);

б) альтернатива Б (0, 2808);

в) альтернатива В (0, 5842).

Альтернатива В имеет преимущество над альтернативой Б, которая в свою очередь имеет преимущество перед альтернативой А.

Из полученных данных, можно сделать вывод, что альтернативы имеют равное значение и отдать небольшое предпочтение можно альтернативам А и В по сравнению с альтернативой Б.

## 4.6 Синтез глобальных приоритетов альтернатив

Для выявления составных, или глобальных, приоритетов методов решения нелинейных уравнений, локальные приоритеты альтернатив располагаются по отношению к каждому критерию. Каждый столбец векторов альтернатив умножается на приоритет соответствующего критерия и результаты складываются вдоль каждой строки [3], формула (4.9):

$$V^{M_j} = \sum_{i=1}^{n} x_i \cdot z_i^{M_j} \tag{4.9}$$

В табл. 4.9 для наглядности представлены исходные данные для расчета значения компонент вектора глобальных приоритетов, полученные на предыдущих этапах МАИ.

Таблица 4.9 – Данные для расчета глобальных приоритетов

| | $x_1(0,0828)$ | $x_2(0,2489)$ | $x_3(0,6259)$ | $x_4(0,0426)$ |
|---|---|---|---|---|
| $z_i^{M_1}$ | 0, 6909 | 0, 4286 | 0, 135 | 0, 135 |
| $z_i^{M_2}$ | 0, 2176 | 0, 1429 | 0, 2808 | 0, 2808 |
| $z_i^{M_3}$ | 0, 0914 | 0, 4286 | 0, 5842 | 0, 5842 |

Рассчитаем по формуле (4.9):

$V^{M_1} = 0,0828 \cdot 0,6909 + 0,2489 \cdot 0,4286 + 0,6259 \cdot 0,135 + 0,0426 \cdot 0,135 = 0,0572 + 0,1067 + 0,0845 + 0,0058 = 0,2542$

$V^{M_2} = 0,0828 \cdot 0,2176 + 0,2489 \cdot 0,1429 + 0,6259 \cdot 0,2808 + 0,0426 \cdot 0,2808 = 0,018 + 0,0356 + 0,1758 + 0,012 = 0,2414$

$V^{M_2} = 0,0828 \cdot 0,0914 + 0,2489 \cdot 0,4286 + 0,6259 \cdot 0,5842 + 0,0426 \cdot 0,5842 = 0,018 + 0,1067 + 0,3657 + 0,0249 = 0,5135$

Для проверки вычислений воспользуемся формулой (4.10):

$$\sum_{i=1}^{n} V^{M_j} = 1 \tag{4.10}$$

$\sum_{i=1}^{3} V^{M_j} = 0,2542 + 0,2414 + 0,5135 = 1,0091$

Рассчитаем погрешность вычислений:

$\delta_V = \frac{1-1,0091}{3} \cdot 100\% = 0,3\%$

Полученная погрешность не превышает допустимые 20%, следовательно, будем считать погрешность удовлетворительной.

Чтобы оценить согласованность всей иерархии, необходимо воспользоваться формулой (4.11):

$$\text{ОС}_{\text{Иерархии}} = \frac{\sum_{i=1}^{n} x_i \cdot \text{ИС}_i}{\text{СС}(m)} \cdot 100\% \tag{4.11}$$

где $x_i$ — значение i-ой компоненты вектора локальных приоритетов второго уровня. $\text{ИС}_i$ — значение i-го индекса согласованности матриц попарных сравнений третьего уровня. СС($m$) — значение случайной согласованности для $m = 3$ [3].

Выполним расчет:

$\text{ОС}_{\text{Иерархии}} = \frac{0,0828 \cdot 0,0267 + 0,2489 \cdot 0,0003 + 0,6259 \cdot 0,0677 + 0,0426 \cdot 0,0677}{0,58} \cdot 100\% = \frac{0,0022 + 0,0001 + 0,0423 + 0,0029}{0,58} \cdot 100\% = \frac{0,0475}{0,58} \cdot 100\% = 8,19\%$

Полученное значение допустимо так как, не превышает 20%.

Результаты вычислений по формуле (4.9) можно рассматривать как значения функции полезности для каждой из альтернатив [3]. Теперь можно оценивать альтернативы по убыванию значений функции полезности.

На первом месте альтернатива В (VMware vSphere), которая «выигрывает» у альтернативы А (KVM):

$(0,5135 - 0,2542) \cdot 100\% = 25,93\%$

Альтернатива А «выигрывает» у альтернативы Б (Hyper-V):

$(0,2542 - 0,2414) \cdot 100\% = 1,28\%$

Далее вычислим вклады критериев в окончательные результаты. Для вычисления $q_i^{M_j}$ («вклада» i-го критерия в значение функции полезности $V^{M_j}$) воспользуемся значениями промежуточных результатов в формуле (4.9) и подставим их в формулу (4.12) [3]:

$$q_i^{M_j} = \frac{x_i \cdot z_i^{M_j}}{V^{M_j}} \cdot 100\%, i = \overline{1,4}, j = \overline{1,3} \quad (4.12)$$

Определим количественные оценки в процентном соотношении вклада каждого из критериев подробнее и сделаем выводы.

Альтернатива А (KVM):

$q_1^{M_1} = \frac{0,0572}{0,2542} \cdot 100\% = 22,5\%$

$q_2^{M_1} = \frac{0,1067}{0,2542} \cdot 100\% = 44,97\%$

$q_3^{M_1} = \frac{0,0845}{0,2542} \cdot 100\% = 33,24\%$

$q_4^{M_1} = \frac{0,0058}{0,2542} \cdot 100\% = 2,28\%$

Исходя из полученных результатов, делаем вывод, что критерий «Масштабируемость» внес наибольший вклад в результат данной альтернативы, так как имеет наивысший приоритет. Так же значительный вес внесли критерии «Отказоустойчивость» и «Цена». Критерий «Интерфейсы управления» не оказал сильного влияния на результат.

Альтернатива Б (Hyper-V):

$q_1^{M_2} = \frac{0,018}{0,2414} \cdot 100\% = 7,45\%$

$q_2^{M_2} = \frac{0,0356}{0,2414} \cdot 100\% = 14,75\%$

$q_3^{M_2} = \frac{0,1758}{0,2414} \cdot 100\% = 72,83\%$

$q_4^{M_2} = \frac{0,012}{0,2414} \cdot 100\% = 4,97\%$

Критерий «Отказоустойчивость» внес наибольший вклад в результат данной альтернативы. Остальные критерий не оказали сильного влияния на результат.

Альтернатива В (VMware vSphere):

$q_1^{M_3} = \frac{0,018}{0,5135} \cdot 100\% = 3,51\%$

$q_2^{M_3} = \frac{0,1067}{0,5135} \cdot 100\% = 20,78\%$

$q_3^{M_3} = \frac{0,3657}{0,5135} \cdot 100\% = 71,22\%$

$q_4^{M_3} = \frac{0{,}0249}{0{,}5135} \cdot 100\% = 4,85\%$

Критерий «Отказоустойчивость» внес наибольший вклад в результат данной альтернативы. Критерий «Масштабируемость» оказал небольшое влияние на результат. Остальные критерий не оказали сильного влияния.

## 4.7 Анализ результатов

Представим все полученные результаты вкладов критериев в виде табл. 4.10.

Таблица 4.10 – Вклады критериев

| | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| $q_i^{M_1}$ | 22,5% | 44,97% | 33,24% | 2,28% |
| $q_i^{M_2}$ | 7,45% | 14,75% | 72,83% | 4,97% |
| $q_i^{M_3}$ | 3,51% | 20,78% | 71,22% | 4,85% |

В результате проведения вариантного анализа определили, что взяв во внимание значения функции полезности для каждой из альтернатив и вклады каждого критерия, наибольшее предпочтение в выборе следует отдать альтернативе B (VMware vSphere), которая «выигрывает» у других альтернатив на 25,93% и 27,21%.

Также стоит упомянуть, что при выборе гипервизора на практике, недостаточно руководствоваться лишь несколькими критериями, так как для каждого конкретного случая необходимо рассматривать довольно обширный список критериев.

# 5 ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

В разделе экспериментальных исследований описано исследование наиболее опасных критических уязвимостей в программном обеспечении, используемом в облачных средах, а также эксплуатация уязвимости CVE-2016-5195 в производственной среде. В подразделе эксплуатации уязвимости также описана защита от уязвимости и меры по предотвращению, мониторингу и оперативному реагированию на подобные инциденты.

## 5.1 Критические уязвимости в 2016 г.

Критические уязвимости 2016 г. в программном обеспечении [18], используемом в облачной среде представлено в табл. 5.1.

Таблица 5.1 – Наиболее опасные критические уязвимости 2016 г.

| CVE ID | CVSS | Тип уязвимости | ПО |
|---|---|---|---|
| CVE-2016-5195 | 7.2 | Получение привилегий | Linux Kernel |
| CVE-2016-6258 | 7.2 | Получение привилегий | Xen |
| CVE-2016-5696 | 5.8 | Получение данных | Linux Kernel |
| CVE-2016-3710 | 7.2 | Запуск кода | QEMU |
| CVE-2016-8655 | 7.2 | DoS, получение привилегий | Linux Kernel |
| CVE-2016-4997 | 7.2 | DoS, получение привилегий, доступ к памяти | Linux Kernel |
| CVE-2016-4484 | 7.2 | Получение привилегий | CryptSetup |
| CVE-2016-6309 | 10.0 | DoS, запуск кода | OpenSSL |
| CVE-2016-1583 | 7.2 | Переполнение стека, получение привилегий, DoS | Linux Kernel |

Рассмотрим подробнее уязвимости из этой таблицы.

Опасная уязвимость CVE-2016-5195 «Dirty COW» обнаружена Филом Остером в ходе исследования зараженного сервера. Данная уязвимость при-

сутствовала в составе ядра Linux 10 лет, начиная с версии 2.6.22. Исправлена в октябре 2016 г [19].

Суть уязвимости состоит в том, что при чтении области данных памяти при использовании механизма copy-on-write (COW), используется одна общая копия, а при изменении данных — создается новая копия, а так называемый «Dirty Bit» указывает был ли изменен соответствующий блок памяти. Проблема возникает при одновременном вызове функции madvise() и записи в страницу памяти, к которой пользователь не имеет доступа на изменение. При многочисленном повторении запросов происходит «гонка» (race condition) и эксплоит получает право на изменение страницы памяти, которая может относится к привилегированному suid-файлу.

Таким образом, с использованием эксплоитов можно повысить привилегии локального пользователя например до пользователя root. В выпущенном патче добавлен соответствующий флаг FOLL_COW, который является индикатором окончания операции COW:

```
diff --git a/include/linux/mm.h b/include/linux/mm.h
+#define FOLL_COW 0x4000  /* internal GUP flag */

diff --git a/mm/gup.c b/mm/gup.c
+static inline bool can_follow_write_pte(pte_t pte, unsigned int
   flags)
+{
+  return pte_write(pte) ||
+  ((flags & FOLL_FORCE) && (flags & FOLL_COW) && pte_dirty(pte));
+}

-if ((flags & FOLL_WRITE) && !pte_write(pte)) {
+if ((flags & FOLL_WRITE) && !can_follow_write_pte(pte, flags)) {

-*flags &= ~FOLL_WRITE;
+*flags |= FOLL_COW;
```

Уязвимость в гипервизоре Xen — CVE-2016-6258 (XSA-182) позволяет привилегированному пользователю гостевой системы выполнить свой код на

уровне хост-системы. Уязвимость применима только к режиму паравиртуализации и не работает в режиме аппаратной виртуализации, а также на архитектуре ARM.

В паравиртуализированных окружениях, для быстрого обновления элементов в таблице страниц памяти пропускались ресурсоемкие повторные проверки доступа. Данные проверки реализовывались с помощью очистки Access/Dirty битов, однако этого оказалось недостаточно [20]. В патче к уязвимости представлен код, в котором исправлены проверки доступов, а именно добавлены дополнительные повторные проверки:

```
diff --git a/xen/include/asm-x86/page.h b/xen/include/asm-x86/page.
   h
+#define _PAGE_AVAIL_HIGH (_AC(0x7ff, U) << 12)

diff --git a/xen/arch/x86/mm.c b/xen/arch/x86/mm.c
+#define FASTPATH_FLAG_WHITELIST \
+(_PAGE_NX_BIT | _PAGE_AVAIL_HIGH | _PAGE_AVAIL | _PAGE_GLOBAL | \
+_PAGE_DIRTY | _PAGE_ACCESSED | _PAGE_USER)

-if ( !l1e_has_changed(ol1e, nl1e, PAGE_CACHE_ATTRS | _PAGE_RW |
   _PAGE_PRESENT) )
+if ( !l1e_has_changed(ol1e, nl1e, ~FASTPATH_FLAG_WHITELIST) )

-if ( !l2e_has_changed(ol2e, nl2e, unlikely(opt_allow_superpage)
-? _PAGE_PSE | _PAGE_RW | _PAGE_PRESENT : _PAGE_PRESENT) )
+if ( !l2e_has_changed(ol2e, nl2e, ~FASTPATH_FLAG_WHITELIST) )

-if ( !l3e_has_changed(ol3e, nl3e, _PAGE_PRESENT) )
+if ( !l3e_has_changed(ol3e, nl3e, ~FASTPATH_FLAG_WHITELIST) )

-if ( !l4e_has_changed(ol4e, nl4e, _PAGE_PRESENT) )
+if ( !l4e_has_changed(ol4e, nl4e, ~FASTPATH_FLAG_WHITELIST) )
```

Уязвимость CVE-2016-5696 в ядре Linux, позволяющая вклиниться в стороннее TCP-соединение была обнародована на конференции Usenix Security Symposium. Из-за недоработки механизмов ограничения интенсивности обра-

ботки ACK-пакетов, существует возможность вычислить информацию о номере последовательности, которая идентифицирует поток в TCP-соединении, и со стороны отправить подставные пакеты, которые будут обработаны как часть атакуемого соединения [21].

Суть атаки заключается в наводнении хоста запросами для срабатывания ограничения в обработчике ACK-пакетов, параметры которого можно получить меняя характер нагрузки. Для атаки необходимо создать шум, чтобы определить значение общего счетчика ограничения интенсивности ACK-ответов, после чего на основании оценки изменения числа отправленных пакетов определить номер порта клиента и осуществить подбор номера последовательности для конкретного TCP-соединения.

В патче к этой уязвимости увеличен лимит TCP ACK и добавлена дополнительная рандомизация для снижения предсказуемости параметров работы системы ограничения ACK-пакетов:

```
diff ––git a/net/ipv4/tcp_input.c b/net/ipv4/tcp_input.c
+int sysctl_tcp_challenge_ack_limit = 1000;
–u32 now;
+u32 count, now;
+u32 half = (sysctl_tcp_challenge_ack_limit + 1) >> 1;

–challenge_count = 0;
+WRITE_ONCE(challenge_count, half +
+prandom_u32_max(sysctl_tcp_challenge_ack_limit));

–if (++challenge_count <= sysctl_tcp_challenge_ack_limit) {
+  count = READ_ONCE(challenge_count);
+if (count > 0) {
+  WRITE_ONCE(challenge_count, count – 1);
```

Уязвимость CVE-2016-3710 «Dark portal» обнаружена в QEMU, который используется для эмуляции оборудования в Xen и KVM. При использовании метода эмуляции stdvga, из-за записи в регистр памяти VBE_DISPI_INDEX_BANK, хранящий смещение адреса текущего банка видеопамяти, возможно обращение к областям памяти, выходящим за границы

буфера, так как предлагаемые банки видеопамяти адресуются с использованием типа byte (uint8_t *), а обрабатываются как тип word (uint32_t *) [22]. Уязвимость позволяет выполнить код на хост-система с правами обработчика QEMU (обычно root или qemu-dm).

В патче к уязвимости добавлены дополнительные проверки диапазона памяти:

```
diff --git a/hw/display/vga.c b/hw/display/vga.c
+assert(offset + size <= s->vram_size);
-if (s->vbe_regs[VBE_DISPI_INDEX_BPP] == 4) {
-  val &= (s->vbe_bank_mask >> 2);
-} else {
-  val &= s->vbe_bank_mask;
-}
+val &= s->vbe_bank_mask;
+assert(addr < s->vram_size);

-ret = s->vram_ptr[((addr & ~1) << 1) | plane];
+addr = ((addr & ~1) << 1) | plane;
+if (addr >= s->vram_size) {
+  return 0xff;
+}
+ret = s->vram_ptr[addr];

+if (addr * sizeof(uint32_t) >= s->vram_size) {
+  return 0xff;
+}

+assert(addr < s->vram_size);
+if (addr >= s->vram_size) {
+  return;
+}

+if (addr * sizeof(uint32_t) >= s->vram_size) {
+  return;
+}
```

Уязвимость в ядре Linux CVE-2016-8655 позволяет злоумышленнику запустить код на уровне ядра с использованием функции packet_set_ring() через манипуляции с кольцевым буфером TPACKET_V3 [23]. Использовать уязвимость можно для выхода за пределы контейнера, для этого необходимо чтобы локальный пользователь имел полномочия по созданию сокетов AF_PACKET.

Устранена уязвимость в декабре 2016 г. с помощью кода, перехватывающий возможные гонки:

```
diff ––git a/net/packet/af_packet.c b/net/packet/af_packet.c
–if (po–>rx_ring.pg_vec || po–>tx_ring.pg_vec)
–  return –EBUSY;

–po–>tp_version = val;
–return 0;
+break;

+lock_sock(sk);
+if (po–>rx_ring.pg_vec || po–>tx_ring.pg_vec) {
+  ret = –EBUSY;
+} else {
+  po–>tp_version = val;
+  ret = 0;
+}
+release_sock(sk);
+return ret;

+lock_sock(sk);
+release_sock(sk);
```

Уязвимость CVE-2016-4997 присутствует в подсистеме netfilter ядра Linux и связана с недоработкой в обработчике setsockopt IPT_SO_SET_REPLACE и может быть использована в системах, использующих изолированные контейнеры [24]. Проблема проявляется при использовании пространств имен для изоляции сети и идентификаторов.

В патче к уязвимости была добавлена дополнительная проверка адреса функции xt_entry_foreach():

```
diff --git a/net/ipv4/netfilter/arp_tables.c b/net/ipv4/netfilter/
   arp_tables.c
+if (pos + size >= newinfo->size)
+  return 0;

+e = (struct arpt_entry *)
+(entry0 + newpos);

+if (newpos >= newinfo->size)
+  return 0;

-if (!mark_source_chains(newinfo, repl->valid_hooks, entry0)) {
-  duprintf("Looping hook\n");
+if (!mark_source_chains(newinfo, repl->valid_hooks, entry0))
-}

diff --git a/net/ipv4/netfilter/ip_tables.c b/net/ipv4/netfilter/
   ip_tables.c
+if (pos + size >= newinfo->size)
+  return 0;

+e = (struct ipt_entry *)
+(entry0 + newpos);
+if (newpos >= newinfo->size)
+  return 0;

diff --git  a/net/ipv6/netfilter/ip6_tables.c b/net/ipv6/netfilter/
   ip6_tables.c
+if (pos + size >= newinfo->size)
+  return 0;

+e = (struct ip6t_entry *)
+(entry0 + newpos);
+if (newpos >= newinfo->size)
+  return 0;
```

Уязвимость CVE-2016-4484 выявлена в пакете CryptSetup, применяемом для шифрования дисковых разделов в Linux. Ошибка в коде скрипта разблокировки позволяет получить доступ в командную оболочку начального загрузочного окружения с правами суперпользователя.

Несмотря на шифрование разделов возможны ситуации при которых некоторые разделы не шифруются (например /boot), таким образом возможно оставить в системе исполняемый файл с правами setuid root для повышения привилегий или скопировать шифрованный раздел по сети для подбора пароля [25].

Для эксплуатации уязвимости необходимо удерживать клавишу Enter в ответ на запрос доступа к зашифрованным разделам. Спустя 70 секунд удерживания клавиши осуществися автоматический вход в командную оболочку.

Проблема вызвана некорректной обработкой лимита на максимальное число попыток монтирования. Исправление доступно в виде загрузки GRUB с параметром «panic» или в виде патча, устанавливающего лимит:

```
diff --git a/scripts/local-top/cryptroot b/scripts/local-top/
   cryptroot
+success=0

+  success=1

-if [ $crypttries -gt 0 ] && [ $count -gt $crypttries ]; then
-  message "cryptsetup: maximum number of tries exceeded for
   $crypttarget"
-  return 1
+if [ $success -eq 0 ]; then
+  message "cryptsetup: Maximum number of tries exceeded. Please
   reboot."
+  while true; do
+    sleep 100
+  done
```

Сотрудники компании Google выявили уязвимость CVE-2016-6309 в OpenSSL, позволяющая злоумышленникам выполнить произвольный код при

обработке отправленных ими пакетов. При получении сообщения размером больше 16 Кб, при перераспределении памяти остается висячий указатель на старое положение буфера и запись поступившего сообщения производится в уже ранее освобожденную область памяти [26].

В патче опубликованном в сентябре 2016 г. исправлена проблема с висячим указателем:

```
diff --git a/ssl/statem/statem.c b/ssl/statem/statem.c
+static int grow_init_buf(SSL *s, size_t size) {
+  size_t msg_offset = (char *)s->init_msg - s->init_buf->data;
+  if (!BUF_MEM_grow_clean(s->init_buf, (int)size))
+    return 0;
+  if (size < msg_offset)
+    return 0;
+  s->init_msg = s->init_buf->data + msg_offset;
+  return 1;
+}

-&& !BUF_MEM_grow_clean(s->init_buf,
-(int)s->s3->tmp.message_size
-+ SSL3_HM_HEADER_LENGTH)) {
+&& !grow_init_buf(s, s->s3->tmp.message_size
++ SSL3_HM_HEADER_LENGTH)) {
```

Ядро Linux подвержено уязвимости CVE-2016-1583, благодаря которой существует возможность поднятия привилегий локальному пользователю при помощи eCryptfs.

С помощью формирования рекурсивных вызовов в пространстве пользователя возможно добиться переполнения стека ядра. Злоумышленник может организовать цепочку рекурсивных отражений в память файла, при которой процесс отражает в свое окружения другие файлы [27]. При чтении содержимого файлов будет вызван обработчик pagefault для процессов, что приведет к переполнению стека.

Соответствующие исправления были приняты в июне 2016 г. в ядре Linux:

```
diff --git a/fs/proc/root.c b/fs/proc/root.c
+sb->s_stack_depth = FILESYSTEM_MAX_STACK_DEPTH;

diff --git a/fs/ecryptfs/kthread.c b/fs/ecryptfs/kthread.c
+#include <linux/file.h>

-goto out;
+goto have_file;

-if (IS_ERR(*lower_file))
+if (IS_ERR(*lower_file)) {
+  goto out;
+}
+have_file:
+  if ((*lower_file)->f_op->mmap == NULL) {
+    fput(*lower_file);
+    *lower_file = NULL;
+    rc = -EMEDIUMTYPE;
+  }
```

## 5.2 Эксплуатация уязвимости ядра Linux CVE-2016-5195

Наиболее опасной уязвимостью 2016 г. является CVE-2016-5195 с кодовым именем «Dirty COW». Такая степень опасности обусловлена тем, что ей подвержены практически все ядра Linux начиная с версии 2.6.22, а также легкостью применения эксплоитов.

В данном разделе применяется эксплоит [28], позволяющий получить права суперпользователя из-под локального пользователя.

Для эксплуатации используется дистрибутив CentOS:

```
# cat /etc/redhat-release
CentOS Linux release 7.2.1511 (Core)
# uname -r
3.10.0-327.el7.x86_64
```

Необходимо повысить привилегии локального пользователя dcow до суперпользователя root, для этого необходимо скачать эксплоит и установить инструменты для компиляции (gcc/g++). После установки всех необходимых настроек компилируем код эксплоита:

```
$ id
uid=1000(dcow) gid=1000(dcow) groups=1000(dcow)
$ git clone https://github.com/gbonacini/CVE-2016-5195
$ cd CVE-2016-5195/
$ make
g++ -Wall -pedantic -O2 -std=c++11 -pthread -o dcow dcow.cpp -lutil
```

После компиляции исходного кода эксплоита, в текущем каталоге появляется бинарный файл, запуск которого в автоматическом режиме позволяет получить права доступа суперпользователя и сменить его пароль на «dirtyCowFun»:

```
$ ./dcow
Running ...
Received su prompt (Password: )
Root password is:   dirtyCowFun
Enjoy! :-)
$ su root
Password: dirtyCowFun
# id
uid=0(root) gid=0(root) groups=0(root)
```

Для исправления уязвимости в кратчайшие сроки для ядра Linux был внесен патч, а мейнтейнеры дистрибутивов опубликовали пакеты с исправлениями. Для обновления дистрибутива и устранения уязвимости необходимо скачать исправленную версию ядра и произвести перезагрузку.

В случае когда целый парк серверов не имеет возможности совершать перезагрузку необходимо использовать такие технологии как kpatch, livepatch, KernelCare. Подобные технологии позволяют применять патчи для ядра Linux без перезагрузки.

Технология KernelCare является коммерческим решением и позволяет без особых проблем в оперативном порядке использовать патчи для ядра, в том числе патч для CVE-2016-5195.

Пример эксплуатации уязвимости при использовании KernelCare:

```
# /usr/bin/kcarectl --uname
3.10.0-327.36.3.el7.x86_64
# /usr/bin/kcarectl --patch-info  | grep CVE-2016-5195 -A3 -B3
kpatch-name: 3.10.0/0001-mm-remove-gup_flags-FOLL_WRITE-games-from-
   __get_user-327.patch
kpatch-description: mm: remove gup_flags FOLL_WRITE games from
   __get_user_pages()
kpatch-kernel: >kernel-3.10.0-327.36.2.el7
kpatch-cve: CVE-2016-5195
kpatch-cvss: 6.9
kpatch-cve-url: https://access.redhat.com/security/cve/cve
   -2016-5195
kpatch-patch-url: https://git.kernel.org/linus/19
   be0eaffa3ac7d8eb6784ad9bdbc7d67ed8e619
$ ./dcow
Running ...
```

Как видно из вывода команд, эксплоит не работает при включенном KernelCare.

Пример отключения KernelCare и попытка эксплуатации уязвимости:

```
# /usr/bin/kcarectl --unload
Updates already downloaded
KernelCare protection disabled, kernel might not be safe
# su - dcow
$ cd CVE-2016-5195/
$ ./dcow
Running ...
Received su prompt (Password: )
Root password is:   dirtyCowFun
Enjoy! :-)
```

В облачной среде, где при каждой минуте простоя поставщик облачных услуг теряет деньги, использование подобных технологий позволяет существенно уменьшить время простоя серверов.

## 5.3 Мониторинг уязвимостей в программном обеспечении

Для своевременного реагирования на уязвимости в ПО, используемом в облачной среде необходим их мониторинг. В ходе исследования различных открытых баз уязвимостей и систем мониторинга не было обнаружено автоматизированных инструментов, позволяющих интегрировать поиск уязвимостей в систему мониторинга.

Для написания такой системы использовалась открытая база уязвимостей сайта www.cvedetails.com [18]. Сайт cvedetails позволяет в ограниченном режиме получить доступ к базе посредством JSON API.

Возможна интеграция программы с любой системой мониторинга или запуском по расписанию. Существует поддержка указания даты поиска уязвимостей, по умолчанию скрипт ищет уязвимости на сегодняшний день:

```
$ ./vulncontrol.py –d 2017–02–18 –m 5
CVE–2017–6074 9.3 http://www.cvedetails.com/cve/CVE–2017–6074/
CVE–2017–6001 7.6 http://www.cvedetails.com/cve/CVE–2017–6001/
CVE–2017–5986 7.1 http://www.cvedetails.com/cve/CVE–2017–5986/
Telegram alert not sent
```

Программа получает данные от сайта в JSON-формате вида:

```
{
    ”cve_id”: ”CVE–2017–5551”,
    ”cvss_score”: ”3.6”,
    ”cwe_id”: ”264”,
    ”exploit_count”: ”0”,
    ”publish_date”: ”2017–02–06”,
    ”summary”: ”The simple_set_acl function in fs/posix_acl.c ...”,
    ”update_date”: ”2017–02–09”,
    ”url”: ”http://www.cvedetails.com/cve/CVE–2017–5551/”
}
```

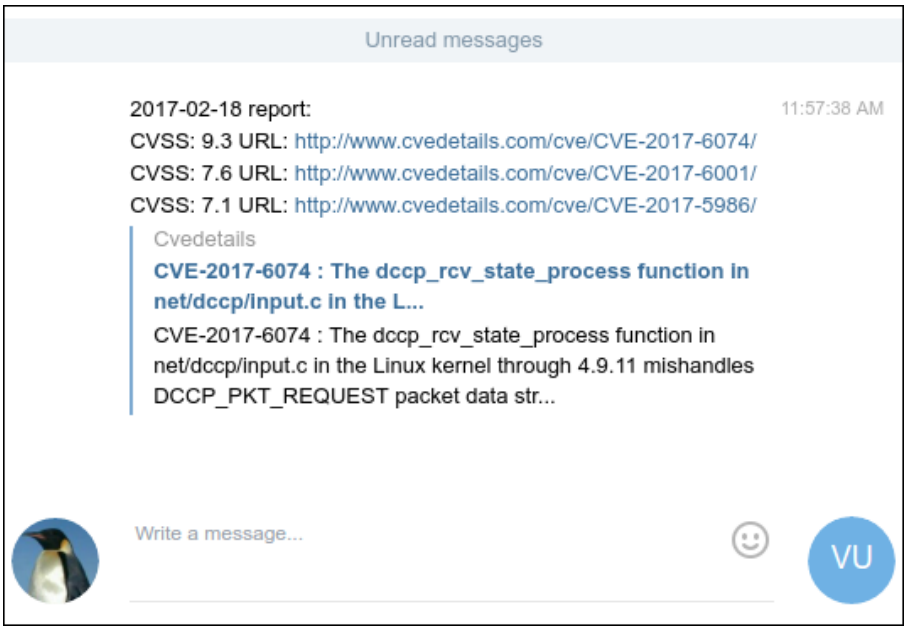


Рисунок 5.1 – Уведомление программы в Telegram

Поддерживается интеграция приложения с мессенджером Telegram (рис. 5.1).

Описание кодов выхода программы представлено в табл. 5.2:

Таблица 5.2 – Коды выхода программы

| Код | Сообщение | Отправлено в Telegram? |
|---|---|---|
| 0 | Уязвимости не найдены | Нет |
| 1 | Уязвимости найдены | Нет |
| 2 | Уязвимости найдены | Да |
| 3 | Уязвимости найдены | Нет, неверные токен и идентификатор |

Исходный код программы представлен в прил. А.

# 6 АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В ходе обзора литературных источников по тематике исследования было рассмотрено понятие облачных вычислений и их развитие, рассмотрены классификации облачных услуг, проанализированы существующие стандарты безопасности и организации, принимающие участие в разработке этих стандартов. Выполнен обзор наиболее известных поставщиков облачных услуг, рассмотрен российский рынок в период с 2014 г. по 2016 г. Составлен список основных угроз облачной безопасности и исследованы тенденции развития облачных вычислений.

В ходе системного анализа была сформирована цель проектирования, разработан список функций проектируемой системы:

- Ф1 — авторизация и аутентификация пользователей;
- Ф2 — сетевая защита;
- Ф3 — идентификация и обработка инцидентов связанных с безопасностью;
- Ф4 — предоставление доступа к услугам;
- Ф5 — мониторинг.

Выделены подсистемы:

а) подсистема аутентификации;
б) подсистема авторизации;
в) подсистема сетевой защиты;
г) подсистема проверки целостности данных.

Определена схема взаимодействия между подсистемами.

В соответствии с принципом функциональности составлена матрица инциденций функций системы и функций назначения подсистем. Произведена декомпозиция подсистем, определены стороны развития системы, определены события и действия, некорректные с точки зрения правил функционирования системы.

В ходе вариантного анализа сравнивались альтернативы гипервизоров (KVM, Hyper-V, VMware vSphere) в соответствии с критериями цены, масштабируемости, отказоустойчивости и интерфейсов управления. Построены матрицы парных сравнений второго и третьего уровня, исследованы согласованности матриц. Синтезированы глобальные приоритеты альтернатив. В результате анализа наибольшее предпочтение решено было отдать альтернативе B (VMware vSphere), однако для более точного определения гипервизора, необходимо сравнивать значительно большее число критериев.

В ходе исследований безопасности облачной среды были сформированы основные проблемы, рассмотрен жизненный цикл данных в облаке, рассмотрена облачная безопасность компании Google.

В ходе экспериментальных исследований были проанализированы уязвимости 2016 г. в программном обеспечении, используемом в облачных вычислениях. Исследованы основные ошибки в программном коде продуктов и способы их исправления.

Эксплуатирована уязвимость CVE-2016-5195, в ходе которой удалось получить права суперпользователя сервера, предложены способы защиты от уязвимостей в ядре Linux.

Для мониторинга уязвимостей была написана программа, анализирующая данные из открытого источника уязвимостей. Данная программа может быть встроена в любую систему мониторинга и имеет возможность уведомлять системного администратора по Telegram.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы магистра были исследованы процессы обеспечения безопасности облачных сред.

В ходе исследования были проанализированы существующие проблемы и стандарты безопасности облачных вычислений, предложены способы решения данных проблем. Рассмотрена специфика предоставления облачных услуг зарубежных и отечественных поставщиков. Проанализированы наиболее опасные уязвимости за 2016 г.

В ходе системного анализа было описано системотехническое представление системы, описаны входные и выходные данные, составлен список функций системы безопасности, произведена декомпозиция системы и описана связь между ее элементами.

В ходе вариантного анализа был произведен сравнительный анализ гипервизоров между тремя альтернативами, в ходе которого был выбран наиболее оптимальный вариант.

В ходе экспериментальных исследования была эксплуатирована уязвимость CVE-2016-5195, благодаря которой локальный пользователь сервера получил доступ к правам суперпользователя. Скрипт не является законченным продуктом и распространяется под свободной лицензией.

Для мониторинга уязвимостей в программном обеспечении облачной среды был разработан скрипт на языке программирования Python. Скрипт осуществляет поиск по открытой базе уязвимостей согласно установленным параметрам. Программа находится в открытом доступе и распространяется под открытой лицензией.

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ЦОД — центр обработки данных

ПО — программное обеспечение

GNU — проект по разработке свободного программного обеспечения

GPL — General Public License, универсальная общественная лицензия

ОЗУ — оперативное запоминающее устройство

SSD — Solid State Drive, твердотельный накопитель

NIST — National Institute of Standards and Technology, Национальный институт стандартов и технологий

СХД — система хранения данных

SaaS — Software as a Service, программное обеспечение как услуга

PaaS — Platform as a Service, платформа как услуга

IaaS — Infrastructure as a Service, инфраструктура как услуга

ОС — операционная система

SOA — service-oriented architecture, сервис-ориентированная архитектура

AWS — Amazon Web Services

GCE — Google Compute Engine

OMG — Object Management Group

ИТ — информационные технологии

CSA — Cloud Security Alliance

DMTF — Distributed Management Task Force

SNIA — Storage Networking Industry Association

OGF — Open Grid Forum

OCC — Open Cloud Consortium

OASIS — Organization for the Advancement of Structured Information Standards

IETF — Internet Engineering Task Force, инженерный совет Интернета

ITU — International Telecommunications Union, Международный институт электросвязи

ETSI — European Telecommunications Standards Institute, Европейский институт телекоммуникационных стандартов

ANSI — American national standards institute, Американский национальный институт стандартов

IDPS — Intrusion Detection and Prevention Systems, руководство по системам обнаружения и предотвращения вторжений

EC2 — Elastic Compute Cloud, веб-сервис компании Amazon, предоставляющий вычислительные мощности в облаке

API — Application Programming Interface, интерфейс создания приложений

vCPU — Virtual Central Processing Unit, виртуальное процессорное ядро

AMI — Amazon Machine Images

EBS — Elastic Block Store, сервис постоянного хранилища блочного уровня для использования с инстансами Amazon

SPI — Secutiry Parameter Index, индекс параметра обеспечения безопасности

BGP — Border Gateway Protocol, протокол граничного шлюза

AS — автономная сетевая система

DNS — Domain Name System, система доменных имен

NTP — Network Time Protocol, протокол сетевого времени

SNMP — Simple Network Management Protocol, простой протокол сетевого управления

NDA — Non-disclosure agreement, соглашение о неразглашении

SSH — Secure Shell, безопасная оболочка

SQL — Structured Query Language, язык структурированных запросов

XSS — Cross-Site Scripting, межсайтовый скриптинг

VPN — Virtual Private Network, виртуальная частная сеть

МАИ — метод анализа иерархии

KVM — Kernel-based Virtual Machine, свободный гипервизор

ИС — индекс согласованности

СС — случайная согласованность

ОС — отношение согласованности

ЛПР — лицо принимающее решение

IoT — Internet of Things, интернет вещей

DDoS — Distributed Denial of Service, распределенная атака на отказ

SDN — Software-defined Networking, программно-определяемая сеть

CVE — Common Vulnerabilities and Exposures, словарь известных уязвимостей

ID — IDentificator, идентификатор

CVSS — Common Vulnerability Scoring System, система оценки уязвимостей

COW — Copy-on-write, копирование при записи

ARM — Advanced RISC Machine, усовершенствованная RISC-машина

GRUB — GRand Unified Bootloader, загрузчик операционной системы

JSON — JavaScript Object Notation, текстовый формат обмена данными

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Прудникова, А.А. Безопасность облачных вычислений / А.А. Прудникова // Мир телекома. – 2013. – №1. – С. 50-55.

2. Методические указания «Процедура системного анализа при проектировании программных систем» для студентов-дипломников дневной и заочной формы обучения специальности 7.091501 / Сост.: Сергеев Г.Г., Скатков А.В., Мащенко Е.Н. – Севастополь: Изд-во СевНТУ, 2005. – 32 с.

3. Методические указания к расчетно-графическому заданию на тему «Метод анализа иерархий» по дисциплине «Теория оптимальных решений» для студентов специальности 7.091501 «Компьютерные системы и сети» дневной и заочной формы обучения / Сост.: Ю.Н. Щепин – Севастополь: Изд-во СевНТУ, 2008. – 28 с.

4. Блюмин С.Л., Шуйкова И.А. Модели и методы принятия решений в условиях неопределенности. – Липецк: ЛЭГИ, 2001. – 138 с.

5. Hogan, M. NIST Cloud Computing Standarts Roadmap / M. Hogan, F. Liu, A. Sokol, J. Tong // NIST Special Publication 500-291, Version 2 Roadmap Working Group, 2013. – 113 с.

6. The 2016 Global Cloud Data Security Study. Ponemon Insitute LLC, 2016. – 40 с.

7. Беккер, М.Я. Информационная безопасность при облачных вычислениях: проблемы и перспективы / М.Я. Беккер, Ю.А. Гатчин, Н.С. Карманов-ский, А.О. Терентьев, Д.Ю. Федоров // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. – 2011. – №1(71). – С. 97-102.

8. Емельянова, Ю.Г. Анализ проблем и перспективы создания интеллектуальной системы обнаружения и предотвращения сетевых атак на облачные вычисления / Ю.Г. Емельянова, В.П. Фраленко // Программные системы: теория и приложения. – 2011. – №4(8) – С. 17-31.

9. Chisnall, D. The Definitive Guide to the Xen Hypervisor / D. Chisnall. – 1st Edition // Prentice Hall Open Source Software Development, 2007. – 320 с.

10. Федеральный закон от 21 июля 2014 г. № 242-ФЗ «О внесении изменений в отдельные законодательные акты Российской Федерации в части уточнения порядка обработки персональных данных в информационно-телекоммуникационных сетях» / Минкомсвязь России // Опубликован 12.02.2016 на официальном интернет-портале Министерства связи и массовых коммуникаций Российской Федерации

11. Облачные сервисы 2016 [Электронный ресурс] // CNews Analytics Режим доступа: https://goo.gl/cmDSMB (Дата обращения: 30.12.2016)

12. Cloud Security Alliance Releases 'The Treacherous Twelve' Cloud Computing Top Threats in 2016 [Электронный ресурс] // Cloud Security Alliance Research Group Режим доступа: https://goo.gl/l2aWLu (Дата обращения: 11.01.2017)

13. ИТ-инфраструктура предприятия 2010: Пути оптимизации [Электронный ресурс] // CNews Analytics Режим доступа: https://goo.gl/jzrrIO (Дата обращения: 05.01.2017)

14. Kaplan, J. Revolutionizing data center energy efficiency / J. Kaplan, W. Forrest, N. Kindler // Technical report, McKinsey & Company, 2008. – 15 с.

15. AWS signature version 1 is insecure [Электронный ресурс] // Daemonic Dispatches Режим доступа: https://goo.gl/70bggH (Дата обращения: 08.02.2017)

16. The CIS Critical Security Controls for Effective Cyber Defense [Электронный ресурс] // SANS website Режим доступа: https://goo.gl/pMjbNE (Дата обращения: 08.02.2017)

17. OWASP Top Ten Project [Электронный ресурс] // OWASP website Режим доступа: https://goo.gl/kSHOjF (Дата обращения: 08.02.2017)

18. CVE security vulnerability database. Security vulnerabilities, exploits, references and more [Электронный ресурс] // CVE Details. The ultimate security

vulnerability datasource Режим доступа: https://goo.gl/I3RtO2 (Дата обращения: 20.02.2017)

19. Dirty COW (CVE-2016-5195) is a privilege escalation vulnerability in the Linux Kernel [Электронный ресурс] // CVE-2016-5195 info website Режим доступа: https://goo.gl/ziy3Nd (Дата обращения: 20.02.2017)

20. Bug 1355987 - (CVE-2016-6258, xsa182) CVE-2016-6258 xsa182 xen: x86: Privilege escalation in PV guests (XSA-182) [Электронный ресурс] // Red Hat Bugzilla Режим доступа: https://goo.gl/dlqtnR (Дата обращения: 20.02.2017)

21. CVE-2016-5696 [Электронный ресурс] // Common Vulnerabilities and Exposures. The Standart for Information Security Vulnerability Names Режим доступа: https://goo.gl/xYpFQQ (Дата обращения: 21.02.2017)

22. CVE-2016-5696 [Электронный ресурс] // Debian Security Bug Tracker Режим доступа: https://goo.gl/BXkTiL (Дата обращения: 21.02.2017)

23. CVE-2016-8655 - Red Hat Customer Portal [Электронный ресурс] // Red Hat Customer Portal Режим доступа: https://goo.gl/QhVbmm (Дата обращения: 21.02.2017)

24. CVE-2016-4997 [Электронный ресурс] // Common Vulnerabilities and Exposures. The Standart for Information Security Vulnerability Names Режим доступа: https://goo.gl/dbtXny (Дата обращения: 21.02.2017)

25. CVE-2016-4484: Cryptsetup Initrd root Shell [Электронный ресурс] // Hector Marco Gisbert - Lecturer and Cyber Security Researcher website Режим доступа: https://goo.gl/Jrfg6H (Дата обращения: 22.02.2017)

26. CVE-2016-6309 : statem/statem.c in OpenSSL 1.1.0a does not consider memory-block movement after a realloc call [Электронный ресурс] // CVE Details. The ultimate security vulnerability datasource Режим доступа: https://goo.gl/JvJPf8 (Дата обращения: 22.02.2017)

27. CVE-2016-1583 [Электронный ресурс] // Debian Security Bug Tracker Режим доступа: https://goo.gl/PIdqGR (Дата обращения: 22.02.2017)

28. gbonacini/CVE-2016-5195: A CVE-2016-5195 exploit example. [Электронный ресурс] // GitHub Режим доступа: https://goo.gl/9tFhHh (Дата обращения: 24.02.2017)

# СПИСОК ИЛЛЮСТРАТИВНОГО МАТЕРИАЛА

# СПИСОК ТАБЛИЧНОГО МАТЕРИАЛА

ПРИЛОЖЕНИЕ А

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

УТВЕРЖДЕН

Программа мониторинга уязвимостей в программном обеспечении

ОПИСАНИЕ ПРОГРАММЫ

Листов — 3

2017

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# https://github.com/Amet13/vulncontrol

from sys import exit
from datetime import datetime
from urllib.parse import urlparse, urlencode
from urllib.request import urlopen
from urllib.error import HTTPError
from json import loads
import argparse

__author__ = 'Amet13'

today = datetime.now().strftime('%Y-%m-%d')

# Arguments parsing
parser = argparse.ArgumentParser()
parser.add_argument('-d', default=today, dest='DATE')
parser.add_argument('-m', default='1', dest='MINCVSS')
parser.add_argument('-t', default='', dest='TGTOKENID', nargs=2)
namespace = parser.parse_args()

try:
    tgtoken = namespace.TGTOKENID[0]
    tgid = namespace.TGTOKENID[1]
except(IndexError):
    tgtoken = ''
    tgid = ''

date = namespace.DATE
mincvss = namespace.MINCVSS
year = date.split('-')[0]
month = date.split('-')[1]

# Array for product IDs
```

```
37 ids = []
38 # Array for results
39 cves = []
40 # Array for Telegram results
41 tgcves = []
42 # Maximum rows for one product
43 numrows = 30
44
45 tgurl = 'https://api.telegram.org/bot'
46 tgfull = '{0}{1}/sendMessage'.format(tgurl, tgtoken)
47 feedlink = 'https://www.cvedetails.com/json-feed.php'
48 source = open('products.txt', 'r')
49 # Getting product IDs from file
50 for line in source:
51     if not line.startswith('#') and line.strip():
52         parsed = urlparse(line)
53         path = parsed[2]
54         pathlist = path.split('/')
55         ids.append(pathlist[2])
56 source.close()
57
58 # Get JSON
59 try:
60     for x in ids:
61         # Link example:
62         # https://www.cvedetails.com/json-feed.php?product_id=47&
              month=02&year=2017&cvssscoremin=10&numrows=30
63         link = '{0}?product_id={1}&month={2}&year={3}&cvssscoremin
              ={4}&numrows={5}' \
64             .format(feedlink, x, month, year, mincvss, numrows)
65         # Going to URL and get JSON
66         getjson = urlopen(link)
67         jsonr = getjson.read()
68         for y in range(0, numrows):
69             try:
70                 jp = loads(jsonr.decode('utf-8'))[y]
71                 if jp['publish_date'] == date:
```

```
72                      result = '{0} {1} {2}' \
73                          .format(jp['cve_id'], jp['cvss_score'], jp
                                ['url'])
74                      tresult = 'CVSS: {0} URL: {1}' \
75                          .format(jp['cvss_score'], jp['url'])
76                      # Keep results in arrays
77                      cves.append(result)
78                      tgcves.append(tresult)
79              except(IndexError):
80                  break
81  except(ValueError, KeyError, TypeError):
82      print('JSON format error')
83
84  # Getting data for Telegram
85  tgdata = '{0} report:\n{1}'.format(date, '\n'.join(tgcves))
86  tgparams = urlencode({'chat_id': tgid, 'text': tgdata}).encode('utf
        -8')
87
88  if len(cves) == 0:
89      print('There are no available vulnerabilities at ' + date)
90      exit(0)
91  else:
92      print('\n'.join(cves))
93      if tgtoken == '' or tgid == '':
94          print('Telegram alert not sent')
95          exit(1)
96      else:
97          try:
98              urlopen(tgfull, tgparams)
99              print('Telegram alert sent')
100             exit(2)
101         except(HTTPError):
102             print('Telegram alert not sent, check your token and ID
                    ')
103             exit(3)
```