МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

Заведующий кафедрой
к.т.н., доцент
М.С. Воробьева

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
магистра

ПОСТРОЕНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДЕЛИ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ КОМПЬЮТЕРНОЙ СЕТИ ДЛЯ ПРОВЕРКИ УМЕНИЯ ПОИСКА И ЭКСПЛУАТАЦИИ УЯЗВИМОСТИ

02.04.03 Математическое обеспечение и администрирование информационных систем
Магистерская программа «Разработка, администрирование и защита вычислительных систем»

Выполнил работу
студент 2 курса
очной формы обучения — Семенов Дмитрий Юрьевич

Научный руководитель
д.п.н., профессор — Захарова Ирина Гелиевна

Рецензент
Доцент кафедры информационной безопасности — Зулькарнеев Искандер Рашитович

Тюмень
2020

ОГЛАВЛЕНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ВУЗ - Высшее Учебное Заведение

Гб - Гигабайт

ГОСТ - Государственный общесоюзный стандарт

ИБ - Информационная Безопасность

Мб - Мегабайт

ОС - Операционная Система

ОЗУ - Оперативное запоминающее устройство

ПО - Программное Обеспечение

СУБД - Система Управления Базами Данных

УВИ - Уязвимая Виртуальная Инфраструктура

ФСТЭК - Федеральная служба по техническому и экспортному контролю

ЦБ РФ - Центральный Банк Российской Федерации

CSRF - Cross-Site Request Forgery

CPU - Central Processing Unit

CTF - Capture The Flag

CVSS - Common Vulnerability Scoring System

CVE - Common Vulnerabilities and Exposures

ISO - International Organization for Standardization

MVC - Model-View-Controller

NAT - Network Address Translation

NVD - National Vulnerability Database

ORM - Object-Relational Mapping

RAT - Remote Access Trojan

SMB - Server Message Block

SQL - structured query language

URL - Uniform Resource Locator

WAF - Web Application Firewall

XSS - Cross-Site Scripting

СПИСОК ТЕРМИНОВ

Аудит: форма независимого, нейтрального контроля какого-либо направления деятельности организации.

Доступностью информации: состояние информации, при котором субъекты, имеющие право доступа, могут реализовать их беспрепятственно.

Информационная инфраструктура: является совокупностью объектов информатизации, обеспечивающая доступ потребителей к информационным ресурсам.

Компьютерная сеть: набор автономных компьютеров, связанных одной технологией.

Компьютерная криминалистика: прикладная наука о раскрытии преступлений, связанных с компьютерной информацией, об исследовании цифровых доказательств, методах поиска, получения и закрепления таких доказательств.

Конфиденциальность информации: свойство информации, указывающее на необходимость ограничения круга субъектов, имеющих доступ к данной информации.

Объект информатизации: совокупность информационных ресурсов, средств и систем обработки информации, используемых в соответствии с заданной информационной технологией, а также средств их обеспечения, помещений или объектов (зданий, сооружений, технических средств), в которых эти средства и системы установлены, или помещений и объектов, предназначенных для ведения конфиденциальных переговоров.

Тестирование на проникновение: метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника.

Угроза: вероятностное событие или процесс, которое посредством воздействия на компоненты информационной системы может привести к нанесению ущерба активам организации, в частности нарушению конфиденциальности, целостности или доступности информации.

Уязвимость: является характеристикой или свойством ИС, использование которой злоумышленником может привести к реализации угрозы.

Целостность информации: свойство информации быть защищенной от несанкционированного искажения, разрушения или уничтожения.

Криптография: наука о способах защиты конфиденциальных данных от нежелательного стороннего прочтения.

Стеганография: наука о скрытой передаче информации путём сохранения в тайне самого факта передачи.

Флаг: информация, которая однозначно подтверждает факт нахождения и эксплуатации уязвимости.

Эксплойт: компьютерная программа, фрагмент программного кода или последовательность команд, которые используются уязвимости в программном обеспечении и применяются для осуществления атаки на вычислительную систему.

Reverse Engineering: исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы.

Reverse shell - Терминальная сессия удаленного управления, инициируемая атакуемым устройством, в результате успешной эксплуатации и внедрения кода.

ВВЕДЕНИЕ

Согласно данным Генеральной прокуратуры РФ, за первую половину 2019 года правоохранительными органами было зарегистрировано увеличение количества преступлений, совершаемых с использованием информационных технологий или в сфере компьютерной информации [Статистический сборник “Состояние преступности в России за июль 2019 г.”]. В связи с этим существует дефицит квалифицированных кадров, занимающихся аудитом информационной безопасности предприятий. Наиболее актуальным навыком таких специалистов является умение поиска и эксплуатации уязвимостей.

Кроме того, с 2017 по настоящее время наблюдается ежегодный рост количества выпускников, обучавшихся по образовательным программам специальностей и направлений подготовки 10.00.00 «Информационная безопасность» (рис. 1). [Первый рейтинг «Подготовка кадров для ИТ-отрасли в регионах»] [Анализ состояния системы подготовки специалистов в области информационной безопасности, с. 5-6]

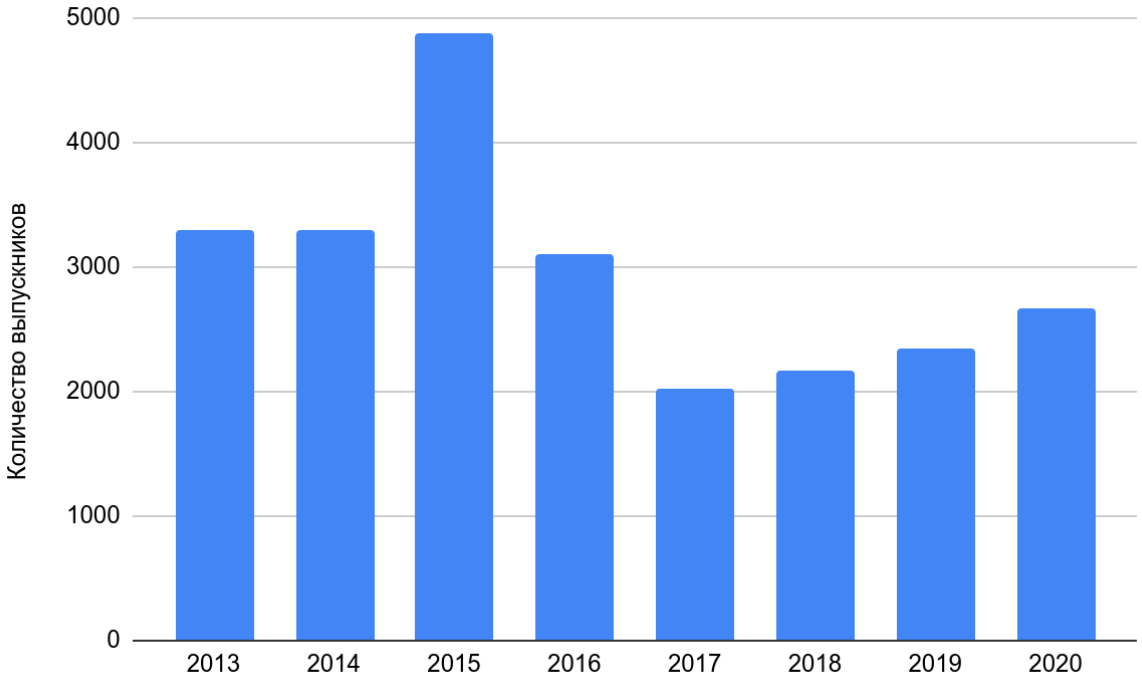

Рис 1. Динамика количества выпускников образовательных организаций по группе специальностей и направлений подготовки 10.00.00 «Информационная безопасность»

Выпускники ВУЗов при этом испытывают недостаток практических навыков, которыми они должны обладать после обучения. Связано это с акцентом на преподавание преимущественно теоретического материала [Пантюхин И.С., Дранник А.Л., Птицын, с. 190-191]. Для решения данной проблемы необходимо разрабатывать занятия на современном оборудовании и ПО, ориентированные именно на практические аспекты ИБ. Однако создание таких заданий для проведения занятий связано с большими затратами труда, времени и ресурсов.

Последующая проверка практических навыков связана с теми же затратами ресурсов. Кроме того, она требует разработки дополнительных заданий, так как проверка знаний на обучающем материале не позволяет достоверно оценить навыки.

Кроме того, как показывают исследования [A virtual environment for the enactment of realistic cyber security scenarios, с. 1-2], компаниям, занимающимся практическими аспектами информационной безопасностями, такими как тестирование на проникновение и аудитом ИБ, приходится затрачивать большое количество времени и ресурсов для создания собственных тестовых лабораторных для оценки навыков поиска и эксплуатации уязвимостей компьютерных инфраструктур.

Сэкономить время в данном случае помогло бы средство, благодаря которому автоматизировано создавались и настраивались виртуальные лабораторные стенды для проверки навыков.

В связи со всем вышеперечисленным, целью данной работы является: разработать программный продукт для автоматизированного создания уязвимых виртуальных инфраструктур с заданными конфигурациями, использование которых позволит сэкономить время и ресурсы компании, а также объективно

оценить практические навыки и умения испытуемого при приеме на работу на должность, связанную с информационной безопасностью.

Для достижения поставленной цели были выделены следующие задачи:

- Исследование особенностей проведения тестирования на проникновение и выделение навыков, которые требуются от специалиста;
- Построение модели уязвимой виртуальной инфраструктуры;
- Исследование существующих разработок в сфере создания виртуальных стендовых лабораторных;
- Исследование особенностей автоматизированного создания виртуальной инфраструктуры и внедрения уязвимостей в них;
- Разработка базового образа операционной системы;
- Разработка базы модулей, содержащих уязвимости, готовых для внедрения в базовый образ;
- Разработка метода внедрения уязвимых модулей в базовый образ;
- Разработка метода создания сети виртуальных машин - виртуальной инфраструктуры;
- Разработка административной панели, которая позволяет производить настройку, запуск и администрирование виртуальной инфраструктуры;
- Тестирование и апробация программного продукта на примере создания наборов инфраструктур с заданными уязвимостями

ГЛАВА 1

В данной главе будут рассмотрены существующие компьютерные инфраструктуры предприятий, широко используемые в современных компаниях, а также относящиеся к ним аспекты информационной безопасности.

Также в данной главе будут рассмотрены подходы к проверке информационной безопасности компаний, а также методы, которые предлагаются для обучения и проверки навыков специалистов. Будут показаны их сильные и слабые стороны, и сделан вывод о необходимости разработки

собственного средства, минимизирующего временные и ресурсные затраты организации на создание виртуальной инфраструктуры.

1.1. ИНФРАСТРУКТУРА КОМПЬЮТЕРНОЙ СЕТИ ПРЕДПРИЯТИЯ

Согласно ГОСТ Р 53114-2008 “Обеспечение информационной безопасности в организации. Основные термины и определения” информационная инфраструктура является набором объектов информатизации, обеспечивающая доступ потребителей к информационным ресурсам [ГОСТ Р 53114-2008, с. 2].

Объект информатизации - Совокупность информационных ресурсов, средств и систем обработки информации, используемых в соответствии с заданной информационной технологией, а также средств их обеспечения, помещений или объектов (зданий, сооружений, технических средств), в которых эти средства и системы установлены, или помещений и объектов, предназначенных для ведения конфиденциальных переговоров [ГОСТ Р 51275-2006, с. 1]. В данной работе не будут рассматриваться помещения и здания, в которых хранятся средства обработки информации, а также места проведения конфиденциальных переговоров.

Под термином компьютерная сеть понимают набор автономных компьютеров, связанных одной технологией. Два компьютера называют связанными, если они в состоянии обмениваться информацией. Сети бывают различных размеров, формы и конфигураций [Э. Таненбаум, Д. Уэзеролл, с. 17].

На основании вышеперечисленного сформируем для дальнейшей работы полное определение информационной инфраструктуры компьютерной сети — это совокупность связанных между собой информационных ресурсов и систем обработки информации, обеспечивающая доступ потребителя к информационным ресурсам.

Характеристиками такой инфраструктуры являются [Урбанович, Романенко, Кабак, с. 45-53]:

- Средства обработки и хранения данных (компьютеры, серверные хранилища). Они, в свою очередь, характеризуются:
    1. Операционной системой;
    2. Установленным программном обеспечением;
    3. Связями с другими устройствами внутри сети.
- Сетевая топология - отражает связность компьютеров между собой. К видам сетевых топологий относятся:
    1. Звезда - топология, где к одному центральному узлу подключены все остальные рабочие станции. Главный узел поддерживает связи между рабочими станциями и разрывает их при необходимости. Наиболее часто встречаемая технология компьютерной сети;
    2. Шина - топология, в которой рабочие станции расположены вдоль одного участка кабеля, называемого сегментом. Наименее распространенный вид топологии;
    3. Кольцо - топология, в которой каждая рабочая станция соединена с двумя другими соседними станциями, образуя кольцо;
    4. Иерархическая топология - топология, в которой каждый узел в иерархии связан с одним узлом выше по иерархии и несколькими более низкого уровня, образуя таким образом комбинацию звезд
    5. Полносвязная топология - топология, в которой рабочая станция связана с каждой другой рабочей станцией.

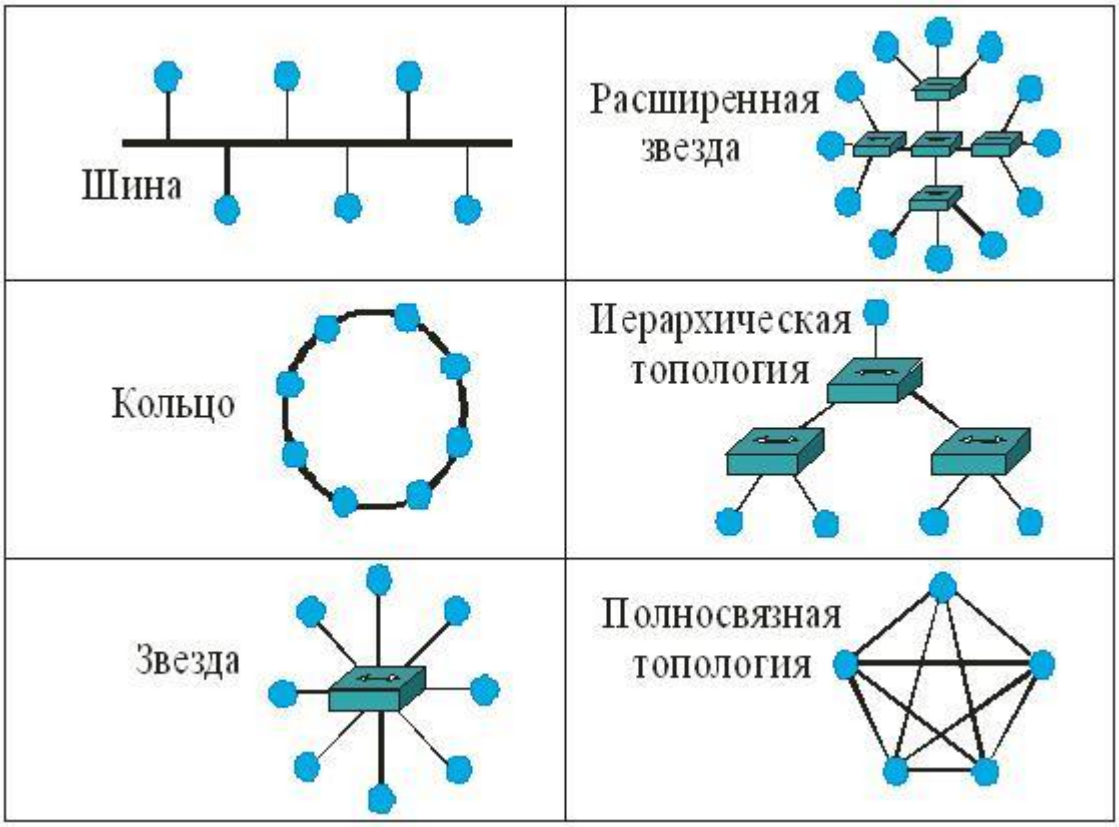

Рис. 2. Топологии сети компьютерной инфраструктуры

1.2. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ КОМПЬЮТЕРНОЙ ИНФРАСТРУКТУРЫ

Согласно ГОСТ Р 53114-2008 “Обеспечение информационной безопасности в организации. Основные термины и определения” информационная безопасность предприятия является состоянием защищенности интересов организации в условиях угроз в информационной сфере. Защищенность достигается обеспечением совокупности свойств информационной безопасности (ИБ) - конфиденциальностью, целостностью, доступностью информационных активов и инфраструктуры организации. Приоритетность свойств информационной безопасности определяется значимостью информационных активов для интересов (целей) организации [ГОСТ Р 53114-2008].

Дадим определения указанным свойствам информации, которые подлежат защите:

- Конфиденциальность - свойство информации, для обеспечения которого необходимо ограничить круга субъектов, имеющих доступ к данной информации [Загинайлов, с. 8];
- Целостность - свойство информации быть защищенной от несанкционированного искажения, разрушения или уничтожения [Певнев, с. 74];
- Доступность - состояние информации, при котором субъекты, имеющие право доступа, могут реализовать их беспрепятственно [Меньших, Константиновна, с. 18].

При нарушении хотя бы одного из данных свойств снижается ценность информации в целом. Возможность нарушения какого-либо из свойств информации в системе называется угрозой.

Угроза представляет собой вероятностное событие или процесс, которое, воздействуя на составляющие информационной системы, может привести к нанесению ущерба активам организации, в частности нарушению конфиденциальности, целостности или доступности информации [Варлатая, Файзенгер, Тимофеева, с. 1]. Уязвимость является характеристикой или свойством ИС, использование которой злоумышленником может привести к реализации угрозы.

Невозможно навсегда и полностью защитить компьютерную инфраструктуру от угроз. Поэтому важно подчеркнуть, что обеспечение информационной безопасности предприятия является не одноразовым действием, обеспечивающим дальнейшую защищенность от всех угроз, а процессом. Следовательно, по истечению определенного срока, заданного законодательным или локальным актом, необходимо проводить проверку защищенности предприятия.

1.3. СПОСОБЫ ПРОВЕРКИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ В КОМПЬЮТЕРНЫХ СЕТЯХ

Согласно постановлению 382-П ЦБ РФ [Положение Банка России от 09.06.2012 N 382-П], проводить оценку защищенности банков необходимо раз в год. Для соответствия другому стандарту России 27001-2006 [ГОСТ Р ИСО/МЭК 27001-2006] не только банкам, но и другим предприятиям необходимо проходить периодический аудит информационной безопасности.

Аудит – форма независимого (с привлечением третьей стороны) контроля и проверки какого-либо направления деятельности организации. Главной целью аудита информационной безопасности является оценка уровня безопасности информационной инфраструктуры компании для управления ИБ в целом с учетом перспектив его развития [Аверченков, с. 18-23]. Также целями проведения оценки защищенности являются:

- Оценка соответствия требований по обеспечению ИБ с предъявляемыми нормативными или локальными актами;
- Проверка соответствия требованиям различных смежных и организаций-партнеров, и при осуществлении правоохранительной деятельности;
- Оценка эффективности системы управления ИБ на объекте ОВД для достижения целей защиты, охраняемой законом информации;
- Определение направлений развития систем защиты конфиденциальной информации для повышения ее уровня.

Существует несколько типов аудита [Лазуткин, с. 2]:

- Внутренний - как непрерывный процесс поддержки защищенности информационной инфраструктуры, которая осуществляется внутренними службами безопасности;
- Внешний - разовое мероприятие оценки защищенности инфраструктуры, проводимое по инициативе руководства компании. В качестве проверяющих выступают сторонние компании, специализирующиеся на проведении таких работ.

Можно выделить следующие основные виды аудита информационной безопасности [Аудит информационной безопасности органов исполнительной власти: учеб. пособие, с. 22-27]:

- Экспертный аудит безопасности - основывается на опыте экспертов, участвующих в процедуре выявления недостатков в системе мер защиты информации;
- Оценка соответствия требованиям руководящих документов ФСТЭК и рекомендациям международного стандарта ISO 17799 [Цуканова, Смирнов, с. 78-80];
- Инструментальный анализ защищенности ИС - использование автоматизированных программных продуктов для выявления уязвимостей программно-аппаратного обеспечения системы;
- Тестирование на проникновение - метод оценки безопасности компьютерных систем или сетей, при котором специалистом используется моделирование действий, совершаемых злоумышленником при попытке взлома. Он позволяет получить объективную оценку того, насколько просто осуществить несанкционированный доступ к информационной инфраструктуре предприятия, а также взглянуть на саму систему с точки зрения злоумышленника и понять, каким способом можно скомпрометировать выбранную систему и какие вредоносные действия на ней можно совершить [Варлатая, Файзенгер, Тимофеева, с. 4];
- Комплексный аудит - включает в себя все вышеперечисленные типы проведения обследования

Наиболее сложным и эффективным способом оценки является тестирование на проникновение, так как он требует непосредственно практических умений взлома компьютерных инфраструктур. Рассмотрим его подробнее.

1.4. ОСОБЕННОСТИ ПРОВЕДЕНИЯ ТЕСТИРОВАНИЯ НА ПРОНИКНОВЕНИЕ

При проведении оценки состояния защищенности информационной инфраструктуры может быть необходимость проведения тестирования на проникновение.

Тестирование на проникновение (penetration testing, pentest, пентест) - метод оценки безопасности компьютерных систем или сетей, при котором специалистом используется моделирование действий, совершаемых злоумышленником при попытке взлома. Он позволяет получить объективную оценку того, насколько просто осуществить несанкционированный доступ к информационной инфраструктуре предприятия, а также взглянуть на саму систему с точки зрения злоумышленника и понять, каким способом можно скомпрометировать выбранную систему и какие вредоносные действия на ней можно совершить [Плешков, Рудер, с. 175].

Существует несколько типов проведения тестирования на проникновение [Бегаев, Бегаев, Федотов, с. 5-9]:

- “Черный ящик” - тестирование проводится в условиях полного отсутствия информации об информационной инфраструктуре предприятия на момент начала тестирования. Например, если это внешнее тестирование методом «черного ящика», исследователю раскрывается только адрес веб-сайта, а задачей в таком случае является осуществление взлома таким образом, как если бы специалист был реальным злоумышленником.
- “Белый ящик” - тестирование проводится в условиях обладания полной информации об информационной инфраструктуре компании и внутренней организации сети. Перед проведением тестирования компания предоставляет схемы сети или список используемых операционных систем и приложений. Хотя в реальной жизни такая ситуация имеет низкую вероятность, метод является наиболее эффективным и точным, так как он является наихудшим сценарием, при котором злоумышленник имеет полное представление о сети.

- “Серый ящика” - в ходе тестирования специалист имитирует действия сотрудника организации. Это означает, что он получает учетную запись для доступа к внутренней сети и имеет стандартные права на доступ и частичное представления об устройстве внутренней инфраструктуры компании, необходимое сотруднику для выполнения своих трудовых обязанностей. Благодаря данному методу возможно оценить внутренние угрозы, которые исходят со стороны сотрудников компании.

Для упрощения определения последовательности действий злоумышленника корпорацией Lockheed Martin была предложена модель Cyber Kill-Chain [Hutchins, Cloppert, Amín, с. 4-7]. Она определяет, какие действия должен совершить злоумышленник для того, чтобы достичь своих целей, атакуя сеть, извлекая данные и поддерживая присутствие в организации.

Опишем этапы, которые входят в данную модель, а также основные инструменты, используемые специалистами на каждом этапе [Yadav, Rao, с. 1-6]:

Первый этап – разведка (Reconnaissance). На данном этапе специалист пытается собрать как можно больше информации из различных, как закрытых, так и открытых, источников о выбранной цели. Разведка может быть:

- Активной - исследователь безопасности использует специальные инструменты, чтобы исследовать целевую сеть и устройства, например, с целью определения диапазона IP-адресов и открытых портов, чтобы определить службы, запущенные на целевых устройствах;
- Пассивной - исследователь безопасности использует доступную любому пользователю сети Интернет информацию для того, чтобы узнать и проанализировать информацию, связанную с технологиями, используемыми в исследуемой организации [Соловьев, Шемяхина, с. 143].

На данном этапе специалист использует инструменты, позволяющие получить дополнительную информацию из публичных источников, например

“Google Dorks”, а также позволяющие перебрать веб-страницы приложения по известным словарям часто встречаемых названий, например “Dirbuster” и “DirSearch”.

Второй этап - сканирование и “вооружение” (Weaponization). После обнаружения служб, исследователь определяет наличие уязвимостей на целевых устройствах.

Для прохождения данного этапа специалисты используют программный продукт “Nmap” для обнаружения открытых портов, сервисов и их версий для дальнейшего анализа на предмет наличия уязвимостей и возможности получения несанкционированного доступа.

Также для “вооружения” используется инструмент “MSVenom”. Данный инструмент позволяет создать специальную программу, вызов которой приведет к вредоносному воздействию на систему и позволит получить доступ к данной системе.

Третий этап - доставка (Delivery). Если доступ к устройству можно получить только посредством использования написанной вредоносной программы (вируса), то происходит “доставка” данного вируса посредством электронной почты, электронных ресурсов и т.д. [Tarnowski, с. 2-7].

Кроме доставки в виде фишингового письма, может быть использован инструмент “Metasploit” для внедрения вредоносного кода и эксплуатации уязвимостей. Данный инструмент также используется на следующих, четвертом и пятом этапах.

Четвертый этап - эксплуатация (Exploit). Доставленный вирус должен быть каким-либо образом вызван (с участием или без участия пользователя целевого устройства) для эксплуатации уязвимости [Kapellas, с. 11-13].

Пятый этап - поддержание доступа (Post-exploitation). В ходе данного этапа осуществляется установка специальных программ (бэкдоров), которые позволяют специалисту в будущем повторно подключаться к системе. Кроме того, на данном этапе исследователь пытается повысить свои права до уровня

администратора, чтобы получить полный контроль над устройством [Spring, Hatleback, с. 3].

На данном этапе может также использоваться инструмент “MSVenom”, совместно с “Metasploit”, но кроме того используются различные дополнительные средства, называемые RAT - средства удаленного доступа.

Шестой этап - наладка связи (Command & control). Исследователь создает скрытый канал связи для обеспечения удаленного незаметного доступа к целевой машине.

Последний этап тестирования - удаление доказательств и следов присутствия специалиста в системе (Exfiltration). Исследователь проверяет, можно ли стереть данные из файлов журналов, хранящих факты, указывающие на его присутствие в сети.

После прохождения всех этапов исследователь может получить доступ к целевому устройству и продолжать исследовать внутреннюю сеть в любое время до его обнаружения.

Достоинствами метода тестирования на проникновение являются:

- высокая достоверность сведении о выявленных уязвимостях благодаря фактическому подтверждению возможности их использования злоумышленником;
- достаточность результатов исследования для оценки критичности выявленных уязвимостей; наглядность получаемых результатов.

К недостаткам методов тестирования на проникновение можно отнести:

- Действия специалиста при проведении тестирования на проникновения плохо поддаются автоматизации [Doupé, Vigna, Cova, с. 18-19], а потому затраты на проведение более высокие, по сравнению с другими способами оценки уровня защищенности;
- Специалист, проводящий тестирование, способен воспроизвести действия только такого нарушителя, который равен ему или уступает по квалификации. По итогу, специалист по тестированию на проникновение

должен обладать высокой квалификацией. Также к недостаткам можно отнести то, что при низкой квалификации специалиста достоверность вывода об отсутствии уязвимостей является низкой.

Соответственно, такое тестирование требует практических навыков и, что тоже немаловажно, опыта взлома реально существующих систем. Следует помнить, что несанкционированный взлом и получение неправомерного доступа к охраняемой законом информации, то есть работа без договора оказаний услуг по тестированию на проникновение, является уголовно наказуемым деянием [Уголовный кодекс Российской Федерации, гл. 28, ст. 272], что усложняет получение опыта работы, особенно у молодых специалистов. Однако теоретические и практические навыки могут быть получены в ходе обучения в университете или на специализированных соревнованиях. Следовательно, возникают необходимость решить следующие проблемы:

1. Если у человека нет навыков тестирования на проникновение, то как легально обучить человека практическим методам пентеста?
2. Если такие навыки, по утверждению самого человека есть, то как проверить его навыки?

Рассмотрим существующие способы обучения и проверки навыков поиска и эксплуатации уязвимостей.

1.5. МЕТОДЫ ОБУЧЕНИЯ И ПРОВЕРКИ НАВЫКОВ ТЕСТИРОВАНИЯ НА ПРОНИКНОВЕНИЕ

На данный момент существует несколько способов обучения и проверки навыков поиска и эксплуатации уязвимостей:

Готовые стендовые лабораторные

Одним из популярных решений является использование заранее созданных лабораторий или виртуальных машин [Кадан, Доронин, с. 6]. Обычно

такие лабораторные доступны для широких масс и каждый может попробовать ее решить.

Существует два вида таких лабораторий:

Онлайн - участнику выдаются настройки для подключения во внутреннюю сеть компании, в которой находится лабораторная, после чего участник может совершать любые действия внутри сети для отработки навыков. Соответственно, каждый участник может видеть изменения, производимые другими участниками и повлиять на них.

Оффлайн - участник загружает на свой компьютер виртуальный образ уязвимой машины, запускает его, производит настройки для получения доступа к машине по сети. После этого уже участник волен совершать любые действия над уязвимым образом для поиска уязвимостей, повышения привилегий и т.д. [Красов, Штеренберг, Москальчук, с. 1-2]

Логично предположить, что оба вида таких лабораторных можно использовать для обучения сотрудников или для проверки навыков при приеме на работу. Однако общим недостатком такого подхода является то, что на данные лабораторные могут быть также доступны ответы. Легкая доступность таких ответов может свести на нет эффективность проверки навыков. Поэтому использование таких лабораторных необходимо сводить к минимуму и исключить повторное использование.

Кроме того, загружаемые уязвимые образы требуют времени на размещение и настройку сети, чтобы хотя бы приступить к решению лабораторной. Также необходима техническая поддержка и администрирование до завершения мероприятия для поддержки доступности уязвимой машины и одинаковых условий участников.

Соревнования по информационной безопасности

Другим способом научить и проверить навыки - участие в соревнованиях по информационной безопасности Capture The Flag (CTF).

Данные соревнования предлагают к решению набор готовых задач разных категорий (Криптография, стеганография, Reverse Engineering, компьютерная криминалистика) и различных уровней сложности [Горбунов, Семакин, Зулькарнеев, с. 1]. Для решения задания необходимо найти “флаг” - данные с ответом, которые достоверно подтверждают, что был использован метод использования.

К плюсам можно отнести использование актуальных уязвимостей, большое количество соревнований различных уровней сложности, а также то, что задания направлены на различные аспекты информационной безопасности. Также согласно исследованиям, участие в CTF позволяет студентам улучшить и закрепить больше знаний более эффективно, в сравнении со студентами, не участвующими в соревнованиях [Chothia, Novakovic, с. 6-9]

Минусами является привязка ко времени соревнований, отсутствие контроля адекватности и сложности заданий при отсутствии контроля за организацией соревнований, сложность отслеживания обмена решениями между участниками.

Разработка собственной стендовой лабораторной

Компания может разработать и свою лабораторную, исходя из опыта команды и требований к соискателю. Главная проблема, с которой сталкиваются создатели подобных проверок - большое количество времени, затрачиваемого на создание и администрирование лабораторной. Для примера, создание огромной публичной лаборатории занимает год у крупного представителя ИБ-компании, а среднее время создание лабораторной для внутреннего пользования занимает 2-3 месяца [A virtual environment for the enactment of realistic cyber security scenarios, с. 1-2].

Также, как и уже готовые лабораторные, кроме затрат на создание, много времени затрачивается на поддержку и администрирование уязвимой машины и сети [A framework for teaching network security in academic environments, с. 4].

К преимуществам данного подхода можно отнести то, что участники своими действиями не наносят вреда реальной инфраструктуре какого либо мероприятия, а могут спокойно экспериментировать, не боясь нанести ущерб компании [Hill, Carver, Pooch, с. 2], а создатели лаборатории точно знают, что в течении некоторого количества времени после создания ответов и решений для созданной лаборатории нет в публичном доступе.

Таблица 1.

Сравнение методов обучения и проверки знаний.

	Соревнования	Готовые лабораторные	Собственные разработки
Большие затраты времени на разработку	Нет	Нет	Да
Необходимо подстраиваться под график проведения	Да	Нет	Нет
Доступны	Нет	Да	Да

ответы на задания			

Все эти способы объединяет необходимость подготовки стендовой лабораторной и заданий, содержащие подготовленные и заранее известные проверяющим уязвимости, поиск и эксплуатация которых и является целью студентов.

В ходе исследования, был сделан вывод, что можно минимизировать время создания новых лабораторных путем автоматизации создания уязвимых виртуальных инфраструктур, которые каждый новый раз содержали бы новый набор уязвимостей. Такой способ избавил бы от необходимости ручного создания лабораторий, минимизировал затраты времени и ресурсов, затрачиваемых на создание виртуальной инфраструктуры при проведении проверки

1.6. СУЩЕСТВУЮЩИЕ РАЗРАБОТКИ

Для начала рассмотрим существующие публичные лабораторные, доступ к которым можно получить в сети интернет:

HackTheBox - онлайн-платформа, которая позволяет пользователям отработать навыки и техники взлома на созданных другими пользователями уязвимых виртуальных устройствах. Для регистрации необходимо пройти испытание, чтобы доказать наличие базовых навыков. Сеть платформы состоит из виртуальных машин с различными операционными системами (семейств Microsoft Windows и Linux) и программным обеспечением. Каждая уязвимая машина оценена по сложности как самим автором, так и пользователями.

Несмотря на разнообразие виртуальных устройств платформы, на ней отсутствуют задания атаки на домен, что не позволяет их отработать [Azam, Beuran, с. 7-9]. Данный аспект является критичным, так как проведение

тестирования на проникновение требует также проверки возможности проведения данной атаки [Azam, Beuran, с. 19].

После 3 месяцев с запуска уязвимой машины, она становится недоступной для всех пользователей. Для получения доступа необходимо оплатить подписку стоимостью 10 долларов в месяц. Также подписка дает доступ к более сложным и комплексным лабораторным.

Кроме того, существует форум, на котором получить различные подсказки, от малейших намеков, которые просто помогут понять в том ли направлении идет исследователь, до определенного пути решения, нахождения и эксплуатации уязвимости.

OSCP labs - Онлайн-платформа американской компании Offensive security, проводящей сертификацию профессиональных специалистов по тестированию на проникновение. Стоимость доступа к данным лабораторным - 800 долларов за 1 месяц, но кроме того, пользователи получают возможность пройти экзамен, подтверждающий необходимые навыки [Schuett, Rahman, с. 12].

Pentestit Lab - Российская бесплатная онлайн-платформа компании Pentestit, обновляемая раз в год. Представляет из себя виртуальную инфраструктуру компьютерной сети предприятия с возможностью проведения сетевых и доменных атак. Компания полностью меняет задания и инфраструктуру каждый год. Подсказки по нахождению уязвимостей можно найти почти сразу после запуска лабораторной, а спустя месяц в общем доступе появляется полное решение [A Cloud-based platform for the emulation of complex cybersecurity scenarios, с. 4-5].

Таблица 2.

Сравнение публичных онлайн лабораторных

Название	Стоимость	Наличие подсказок	Наличие решения	Наличие атак на домен
HackTheBox	Бесплатно + подписка	Есть	Есть	Нет
OSCP Labs	$800	Есть	Нет	Есть
Pentestit	Бесплатно	Есть	Есть	Есть

Кроме данных онлайн-платформ существует программная разработка, позволяющая автоматизировано создавать уязвимые виртуальные машины «Security Scenario Generator (SecGen)» [Hackerbot: Attacker Chatbots for Randomised and Interactive Security Labs, с. 2]. Данная разработка использует ограниченный набор заранее известных уязвимостей, различная комбинация которых приводит к созданию каждый раз новых уязвимых виртуальных машин.

Одной из особенностей данной разработки является возможность эмуляции действий злоумышленника для отработки навыков предотвращения вторжений и расследования инцидентов ИБ [Schreuders, Ardern, с. 1-2].

Недостатком данного решения является отсутствие возможности создания инфраструктуры компьютерной сети и, как следствие, отсутствие возможности проведения сетевых и доменных атак. В рамках данной работы более критичным является именно возможность создания виртуальной инфраструктуры компьютерной сети предприятия, поэтому, ввиду отсутствия программного средства, полностью удовлетворяющего этому требованию, возникает необходимость создания такого средства. Для этого необходимо сначала рассмотреть и проанализировать существующие виды и списки уязвимостей и выработать концепт программного продукта и метод внедрения уязвимости в виртуальную машину.

1.7. АНАЛИЗ И СРАВНЕНИЕ СПИСКОВ УЯЗВИМОСТЕЙ

Для разработки метода внедрения уязвимостей были проанализированы существующие открытые базы уязвимостей, классификации угроз в них и их характеристики. Для анализа были выбраны следующие базы [Tierney, с. 7-9]:

- CVE - одна из самых старых баз, первые уязвимости которой датируются 1998 годом. Каждая уязвимость содержит кодовый номер, описание, компанию, в чьем ПО была найдена уязвимость и дату создания записи.
- NVD - Национальная база уязвимостей США. Содержит описание уязвимости, ссылку на публичные источники, сообщившие о найденной уязвимости, а также известные названия и версии ПО, которые могут быть подвержены уязвимости. Существует исследование возможности использования данной базы для автоматической оценки качества локальных предписаний безопасности [Vulnerability analysis For evaluating quality of protection of security policies, с. 1-2].
- OSVDB - другая открытая база уязвимостей, по содержанию похожая на NVD, но при этом также содержит и описание метода исправления уязвимости [Федорченко, Чечулин, Котенко, с. 4].
- База ФСТЭК - Российская база уязвимостей. Отличительной чертой является огромное количество фильтров, по которым можно находить и группировать уязвимости. Каждая запись об уязвимости содержит информацию об описании уязвимости, вендоре ПО, версии ПО, в которых обнаружена уязвимость, тип ошибки, класс уязвимости, дату выявления, уровень опасности согласно методу оценки CVSS [Houmb, Franqueria, Engum, с. 3], способ эксплуатации, а также способ устранения уязвимости.

Главными компонентами для нашей задачи является наличие в базе названия операционной системы, вендора и версий ПО, класса уязвимости, а также оценки CVSS. Оценка CVSS позволяет узнать “сложность” эксплуатации той или иной уязвимости, что позволит настраивать сложность виртуальной

инфраструктуры [Автоматизированная технология сопоставления угроз и уязвимостей безопасности информации в информационных системах, с. 2-3].

Таблица 3.

Сравнение баз уязвимостей

	CVE	NVD	OSVD	ФСТЭК
Описание уязвимости	Да	Да	Да	Да
Вендор ПО	Да	Да	Да	Да
Версия ПО	Да	Да	Да	Да
Операционная система	Нет	Нет	Нет	Да
Тип ошибки	Нет	Да	Нет	Да
Класс уязвимости	Нет	Нет	Нет	Да
Способ устранения	Нет	Нет	Нет	Да
Оценка CVSS	Нет	Да	Да	Да

Исходя из результатов анализа, можно сделать вывод о том, что наиболее эффективным будет использование базы уязвимостей ФСТЭК в связи с тем, что она представляет собой наиболее подробную базу, содержащую всю нужную информацию.

Отдельно стоит выделить базу exploit-db. В данной базе, кроме описаний уязвимостей содержится и публично открытый программный код эксплуатации уязвимостей. Комбинация данной базы с базой ФСТЭК может быть

необходимым при создании дополнительной функции автоматических проверок успешного внедрения уязвимостей.

Также необходимо отметить рейтинг уязвимостей “OWASP TOP-10”, созданный некоммерческой организацией OWASP. На основании собранных ими данных они составили рейтинг наиболее часто встречаемых уязвимостей. На данный момент последним актуальным списком является документ 2017-ого года, в котором указаны следующие уязвимости [Wichers, Williams, с. 7-18]:

1. Внедрение кода (code injection);
2. Некорректная аутентификация и управление сессией;
3. Межсайтовый скриптинг (XSS);
4. Нарушение контроля доступа;
5. Небезопасная конфигурация;
6. Утечка чувствительных данных;
7. Недостаточная защита от атак;
8. Подделка межсайтовых запросов (CSRF);
9. Использование компонентов с известными уязвимостями;
10. Недостаточное журналирование и мониторинг.

Данные уязвимости необходимо учитывать и отдавать им приоритет при разработке модулей внедряемых уязвимостей.

1.8 ВЫВОДЫ ПО ГЛАВЕ 1

В данной главе были рассмотрены существующие компьютерные инфраструктуры предприятий, широко используемые в современных компаниях, на базе которых была выработана модель виртуальной инфраструктуры.

Также в данной главе были рассмотрены подходы к проверке информационной безопасности компаний, а также методы, которые предлагаются для обучения и проверки навыков специалистов. Проанализировав их был сделан вывод о больших затратах времени и ресурсов компании, затрачиваемых на их использование, и необходимости разработки собственного

средства, минимизирующего данные затраты организации на создание виртуальной инфраструктуры.

ГЛАВА 2

В данной главе рассматриваются роли пользователей разрабатываемого программного продукта. На основании ролевой модели и информации, указанной в первой главе, будут выработаны требования, которые предъявляются к данному программному продукту.

Также в данной главе рассматривается архитектура разрабатываемого программного продукта, выбор используемых технологий. Рассматривается выбор и настройка базовых образов, на базе которых будут реализовываться виртуальные машины в составе виртуальной инфраструктуры, а также описаны разработанные модули, содержащие основные уязвимости согласно рейтингу OWASP TOP-10, и административная панель, позволяющая производить настройку, запуск и администрирование инфраструктуры.

2.1. ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОМУ ПРОДУКТУ

При разработке ПО необходимо учесть, разделить пользователей на группы по ролям:

Оператор - пользователь, производящий настройку, генерацию и проверку виртуальной инфраструктуры. Имеет доступ ко всем возможностям ПО. На основании данной роли к разрабатываемому продукту предъявляются следующие требования:

- ПО должно генерировать виртуальные машины на основании заранее заданных оператором данных;
- В ПО необходимо предусмотреть возможность включения сторонних виртуальных машин по желанию оператора;
- ПО должно иметь административный интерфейс для создания виртуальной инфраструктуры. Доступ к данной панели должен быть только у оператора;
- ПО должно иметь функционал перезапуска созданных виртуальных машин;

Проверяющий - пользователь, имеющий доступ к проверочным значениям. На основании данной роли к разрабатываемому продукту предъявляются следующие требования:

- ПО должно иметь функционал проверки информации (“флага”), добытой испытуемым в ходе поиска и успешной эксплуатации уязвимости;
- ПО должно иметь функционал предоставления всех флагов, находящихся в инфраструктуре.

Испытуемый - пользователь, имеющий доступ только к виртуальным машинам и системе проверки флага, но не имеющий доступа к системе создания и настройки виртуальной инфраструктуры. На основании данной роли к разрабатываемому продукту предъявляются следующие требования:

- Запуск ПО должен быть возможен без дополнительной настройки компьютера

Данные роли могут быть совмещены и сокращены до двух пользователей: оператор, который также выполняет функции проверяющего и испытуемый.

Кроме требований, предъявляемых на основании ролевой модели, на основании пунктов, перечисленных в 1 главе, разрабатываемое ПО должно соответствовать следующим требованиям:

1. Для генерации виртуальных машин должны использоваться собственные заранее разработанные модули;
2. Генерация виртуальной машины, содержащей модуль уязвимости, должна происходить на базе основного образа виртуальной машины, содержащем минимально необходимый набор программного обеспечения;
3. ПО должно иметь функционал объединения нескольких сгенерированных или добавленных виртуальных машин в сеть, выполнение которых было бы возможно только “по цепочке”, одну за другой;
4. ПО должно генерировать случайное значение, предоставляемое испытуемым для подтверждения нахождения и эксплуатации уязвимости.

Данное значение быть предоставлено проверяющему и держаться в секрете от испытуемого;

5. В состав ПО должны входить модули, которые позволяют провести проверку навыков, необходимых для поиска наиболее часто встречаемых уязвимостей согласно рейтингу “OWASP TOP 10”.

2.2. МОДЕЛЬ УЯЗВИМОЙ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ

В основе уязвимой виртуальной инфраструктуры лежит объектная модель, в составе которой находятся:

- Конечное устройство - представляет собой имитацию компьютера пользователя или сервера во внутренней сети компании. Моделируется при помощи технологий виртуализации на базе заранее разработанного базового образа. В рамках данной модели предусмотрено два класса:
    - Конечное устройства - моделирует непосредственно действия компьютеров и серверов.
    - Сетевое устройство - моделирует действия устройств маршрутизации. Необходимость данного класса обуславливается тем, что для дополнительных настроек конфигурации сетевой связи между конечными устройствами и достижения заданных топологий сети.

  В разрабатываемом программном продукте конечное устройство представлено виртуальной машиной.
- Уязвимость - представляет собой ошибку логики программы, благодаря которой возникает ситуация, при которой возможно получить несанкционированный доступ к информации.

  В разрабатываемом программном продукте конечное устройство представлено сторонним программным обеспечением, его настройками и информацией, которую необходимо найти и предоставить для подтверждения эксплуатации уязвимости.

- Топология сети - представляет собой то, как соединены компьютеры между собой. Примеры существующих топологий сети представлены в главе 1.1. Определенные топологии достигаются путем комбинации сетевых устройств, их настроек и настроек интерфейсов виртуальных машин.

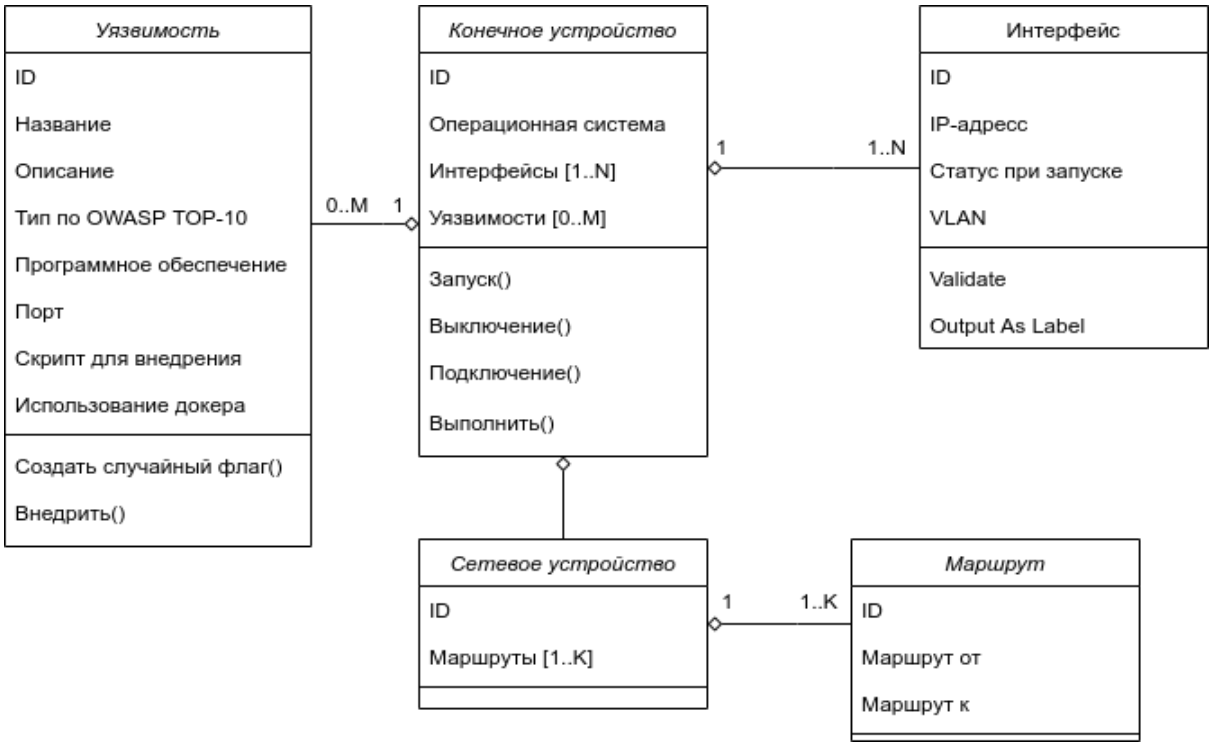

Рис. 3. Объектная модель уязвимой виртуальной инфраструктуры

Программный код классов, представляющих собой объекты модели, представлен в приложении 1.

2.3. АРХИТЕКТУРА ПРОГРАММНОГО ПРОДУКТА АВТОМАТИЗИРОВАННОГО СОЗДАНИЯ УЯЗВИМОЙ ВИРТУАЛЬНОЙ ИНФРАСТРУКТУРЫ

На основании изученного материала и требований была предложена архитектура, на базе которой реализован программный продукт автоматизированного создания уязвимой виртуальной инфраструктуры (рис.4).

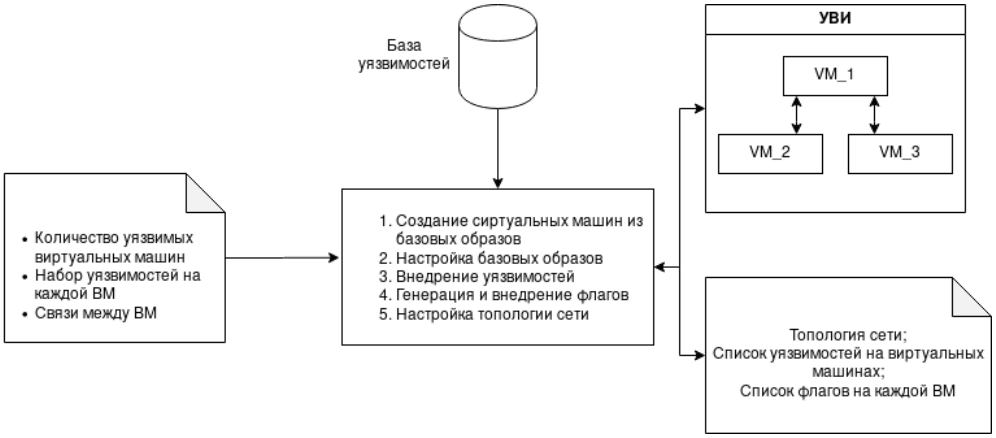

Рис. 4. Архитектура программного продукта

В качестве входных данных программе подаются характеристики системы желаемой виртуальной инфраструктуры уязвимой компьютерной сети, выделенные выше, а именно: количество виртуальных машин в инфраструктуре, операционные системы на них, набор уязвимостей, который должен быть на этих машинах, конфигурация сети.

В процессе работы программа на основании заранее разработанного базового виртуального образа конечного устройства, не содержащего уязвимостей, создает виртуальные машины согласно количеству, указанному оператором при настройке УВИ.

Дальнейшее внедрение уязвимости производится при помощи программного кода доставки определенной уязвимости на систему. Конкретный алгоритм для каждой уязвимости загружается из базы данных уязвимостей. Также на данном этапе производится генерация случайных флагов и их запись для дальнейшей проверки.

Далее программным продуктом настраивается виртуальная сеть между созданными уязвимыми машинами, что и является виртуальной инфраструктурой, к которой проверяющей должен выдать доступ испытуемым.

Кроме того, в качестве дополнения к созданной системе, оператору и проверяющему выдается отдельный список с перечислением конкретного местонахождения уязвимостей на виртуальных машинах, топологии

виртуальной сети и информации, которую необходимо получить участникам для подтверждения навыков поиска и эксплуатации уязвимостей.

Окончательный выбор метода проверки знаний остается за проверяющим. Предлагается два способа:

1. Проверка на основании добытой информации. По аналогии с соревнованиями, подтверждением получения доступа к системе будет информация, к которой открывается доступ при эксплуатации определенной уязвимости. Используя автоматические системы проверки можно еще больше упростить проверку знаний, где итоговая оценка будет выставляться исходя из отношения количества найденной информации к ненайденной.
2. Проверка на основании отчетов. Это более детальная проверка, требующая от участника составления отчета по найденным уязвимостям, а от проверяющего - ручной проверки данного отчета. Несмотря на затраты времени, данная проверка является наиболее приближенной к реальной работе, где помимо самого поиска уязвимостей, важной частью является составление отчета по проведенной работе. Потому это наиболее полный способ проверки при поиске соискателя на позицию аудитора систем информационной безопасности.

2.4. ВЫБОР ТЕХНОЛОГИИ ВИРТУАЛИЗАЦИИ

Виртуальная машина (ВМ) представляет собой эмуляцию аппаратного обеспечения заданной платформы, а также изолирующей программы и операционные системы друг от друга.

К основным функциям виртуальных машин относятся [Вяткин, Никитина, с. 2]:

- ограничение возможности и доступа программ, а также защита информации;
- эмуляция различных архитектур и проверка их производительности;
- оптимизация используемых ресурсов;

- исследование работы и поведения программ;
- упрощение переноса конфигураций ОС;
- тестирование и отладка ПО.

Для создания и запуска виртуальной машины на компьютере используется гипервизор. Под гипервизором понимают специальную программу или аппаратную схему, позволяющая нескольким операционным системам выполняться одновременно. Именно он обеспечивает изоляцию процессов и приложений друг от друга [Szefer, Lee, с. 437]

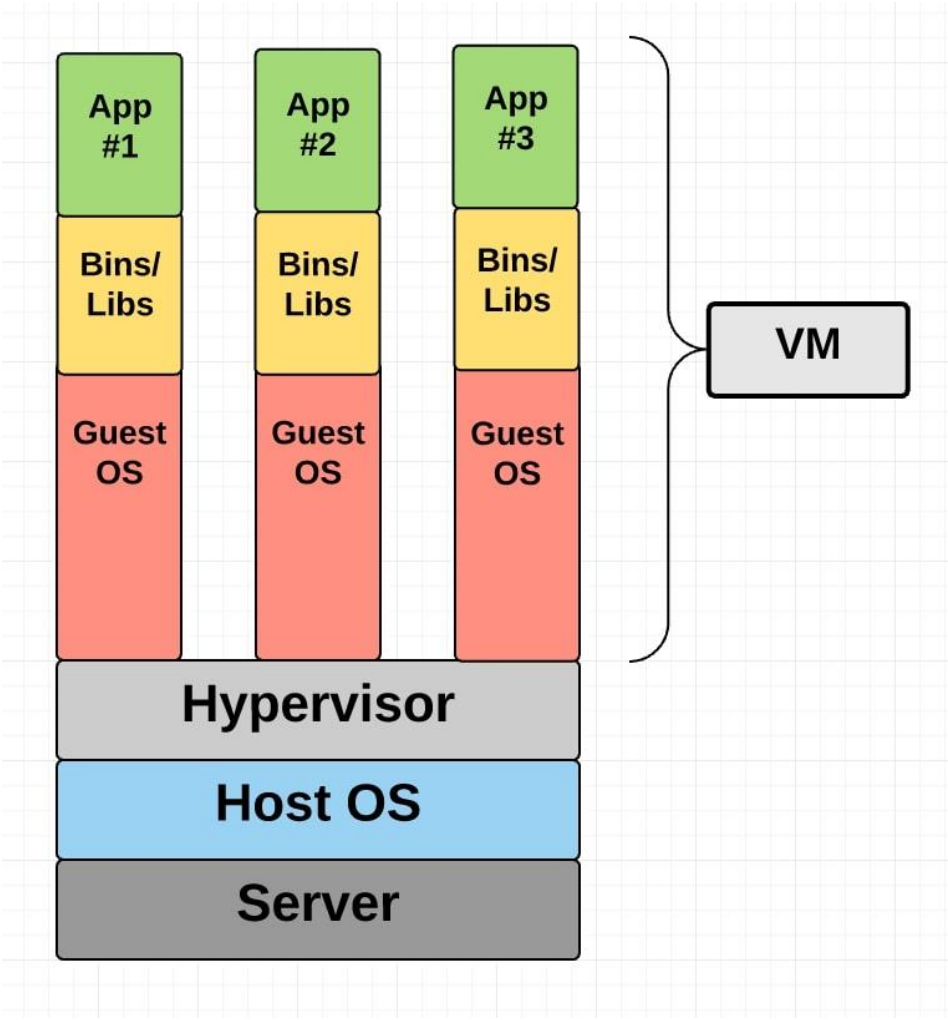

Рис. 5. Схема работы гипервизора

Для создания таких виртуальных машин необходимо выбрать технологию виртуализации, которая позволяла бы решать поставленные задачи, а также

соответствовала требованиям. Для сравнения технологий были выбраны такие программные средства, как VirtualBox, VMware Workstation Player [Алексеев, Красноперова, Вахрушева, с. 121-128.].

VirtualBox - бесплатный программный продукт компании Oracle, используемый для виртуализации.

VMware Workstation Player - бесплатный, при условии некоммерческого использования, программный продукт, на основе виртуальной машины VMware Workstation. В связи с бесплатностью, обладает ограниченной функциональностью и предназначен для запуска образов виртуальных машин, созданных в других продуктах компании VMware.

Сравнение обоих продуктов представлено в таблице 4.

Таблица 4.

Сравнение технологий виртуализации

	VirtualBox	VMware
Стоимость лицензии	Бесплатно	Платно 250$ / бесплатно при ограниченной функциональности
Размер устанавливаемого ПО	63 Мб	463 Мб
Поддержка хостовых ОС	Linux, Windows, MacOS, Solaris, FreeBSD	Linux, Windows, MacOS
Поддержка гостевых ОС	Linux, Windows, MacOS, Solaris, FreeBSD	Linux, Windows, MacOS, Solaris, FreeBSD
Поддержка	VDI, VMDK, VHD	VMDK

виртуальных жестких дисков		
Поддержка аппаратной виртуализации	Да	Да
Поддержка программной виртуализации	Да	Нет
Интерфейс удаленного доступа	PHPVIRTUALBOX	Нет
Поддержка сетевых систем хранения данных	ISCSI, NFS, SMB	Нет
Поддержка снапшотов	Да	Да

Так как в рамках данной работы создается не одна виртуальная машина, а сеть (виртуальная инфраструктура), то важно рассмотреть сетевые возможности представленных технологий виртуализации. Для работы с сетевыми настройками каждая из технологий виртуализации предлагает свои виртуальные сетевые адаптеры. Для каждого такого адаптера можно настроить три типа

- NAT - настройка, при которой IP-адрес прячется за IP-адресом хостовой системы. Доступ к виртуальной машине с такими настройками можно настроить путем проброса портов, то есть настроив хостовой компьютер таким образом, чтобы его определенные порты перенаправляли на гостевую машину.
- Bridge - настройка, при которой виртуальная машина подключается к локальной сети напрямую, используя реальный сетевой адаптер хостового компьютера.

- Host-only - настройка, при которой виртуальная машина может взаимодействовать только с хостовым компьютером.
- Internal-network - настройка, при которой IP-адрес машины не транслируется в публичную сеть, а сетевой взаимодействие с виртуальной машиной быть только с другим виртуальным компьютером, находящимся в той же виртуальной сети.

Также для данных настроек характерны определенные режимы работы, местами отличающиеся в реализации у VirtualBox и VMware. Информация об этому указана в таблице №5.

Таблица 5.

Режимы работы сетевых настроек

Направление передачи	NAT (VirtualBox)	NAT (VMware)	Bridged	Internal network	Host-only
От ВМ к Хостовой ОС	Да	Да	Да	Нет	Да
От Хостовой ОС к ВМ	Проброс портов	Да	Да	Нет	Да
От ВМ к ВМ	Нет	Да	Да	Да	Да
От ВМ к физ. лок. сети	Да	Да	Да	Нет	Нет
От физ. лок. сети к ВК	Проброс портов	Проброс портов	Нет	Нет	Нет

Проанализировав данные, представленные в таблицах №4 и №5, был сделан вывод об использовании программного обеспечения VirtualBox ввиду его

бесплатности, больших возможностях работы с различными гостевыми и хостовыми ОС, а также большим количество виртуальных жестких дисков и сетевых систем хранения данных, что позволит охватить больше технологий и лучшую переносимость разрабатываемого программного продукта.

Наряду с виртуализацией виртуальных машин выделяют контейнеризацию. Основным отличием двух этих технологий является то, что контейнеризация изолирует не аппаратное обеспечение компьютера, а эмулируемые программные процессы от основных процессов ОС. Рассмотрим основные системы контейнеризации:

LXC - система контейнеризации на уровне ОС, предназначенная для запуска нескольких изолированных экземпляров ОС Linux на одном устройстве.

Docker - ПО для автоматизации развёртывания и управления приложениями на основе контейнеризации.

Особенностями контейнеризации является производительность, масштабируемость, легкость в обслуживании и быстрое развертывание приложений [Артамонова, с. 8-10]. Сравнивая LXC и Docker, можно прийти к выводу о схожести функциональных возможностей: оба приложения поддерживают изоляцию процессов, файловой системы и сети, осуществляют возможность сохранения и восстановления процессов, а также возможность “живой” миграции [Адрова, Полежаев, с. 4]. Однако, ввиду большей популярности, доступности большего количество образов и простоты использования, для дальнейшей работы будет использовать ПО Docker [Шафигуллина, с. 51].

В рамках данной работы мы рассматриваем возможности использования как ПО виртуализации VirtualBox, так и контейнеризации Docker. Также, так как некоторые уязвимые модули, о которых будет сказано ниже, используют контейнеризацию, необходимо предусмотреть возможность использования контейнеров Docker не только как самостоятельное решение, но и внутри базового образа.

Для работы с виртуальными машинами существует открытое API VirtualBox, взаимодействие с которой осуществляется посредством команд консольного терминала ОС. Автоматизированная сетевая настройка, посредством таких команд является нетривиальной задачей, помочь решить которую призвано ПО Vagrant.

Шаблон конфигурационного файла, необходимого для запуска виртуальных машин находится приложении 2.

2.5. РАЗРАБОТКА БАЗОВЫХ ОБРАЗОВ КОНЕЧНЫХ УСТРОЙСТВ

Согласно требованиям, предъявленным к разработке, выбранный в качестве базового должен обладать минимальным необходимым набором ПО. К возможностям, которые должны быть возможны относятся:

- базовый образ должен быть создан на одном из дистрибутивов Linux;
- возможность получения непосредственного доступа к виртуальной машине посредством протокола SSH;
- возможность создания нескольких пользователей, разделенных по уровням привилегий;
- возможность администрирования сетевых настроек;
- наличие системы логирования для поиска и исправления возможных ошибок;
- поддержка виртуализации Docker;
- поддержка ПО Vagrant для обеспечения автоматизации сетевых настроек виртуальной инфраструктуры.

Для сравнения и выбора операционной системы базового образа были выбраны:

- Debian - наиболее популярный дистрибутив, используемый как для серверов, так и для рабочих станций;
- OpenSUSE - дистрибутив, основным отличием которого является собственная система настройки и администрирования YaST;

- CentOS 7 - дистрибутив, основанный на коммерческом Red Hat Enterprise Linux компании Red Hat и совместимый с ним;
- Alpine Linux - дистрибутив, ориентированный на безопасность, легковесность и минимальную требовательность к компьютерному железу. Одной из отличительных черт является наличия установочного образа специально настроенного для разворачивания на виртуальных машинах типа VirtualBox.

Сравнение операционных систем представлено в таблице №6.

Таблица 6.

Сравнение дистрибутивов Linux.

	Debian	OpenSUSE	CentOS 7	Alpine linux
Занимаемое место на диске	940 Мб	700 Мб	1.1 Гб	130 Мб
Количество ПО в официальных репозиториях	68000	34000	17565	8135
Минимальные требования к процессору	1 GHz	1,6 GHz	1 GHz	1,5 GHz
Минимальные требования к оперативной памяти	512 Мб	1 Гб	512 Мб	256 Мб

Минимальные требования к жесткому диску	10 Гб	3 Гб	1 Gb	130 Мб

Как видно из сравнения, наименее требовательным к компьютерным ресурсам является дистрибутив Alpine linux. К его минусам можно отнести малое количество ПО в официальных репозиториях, но данная проблема решается добавлением в инструмент по управлению пакетам дополнительных репозиториев, поддерживаемых сообществом.

Данный дистрибутив был установлен на виртуальную машину, который было выделено 1 Гб ОЗУ, 1 ядро процессора и динамически увеличивающийся виртуальный жесткий диск, максимальной емкостью 8 Гб.

Дополнительно на базовый образ для соответствия требованиям были установлены ПО Docker, текстовый редактор Vim, утилита Iptables для управления работой межсетевого экрана.

Фактический занимаемый на жестком диске объем памяти созданным базовым образом является 505 Мб.

Кроме того, для реализации сетевых функций, а также требования по объединению нескольких виртуальных машин в сеть и решение их по цепочке было решено создать и использовать еще один базовый образ, эмулирующих операционную систему RouterOS, использующуюся на сетевом оборудовании компании Mikrotik.

Использование данной ОС обусловлено тем, что её использование бесплатно, она доступна в открытом доступе, а также возможно использование на технологии виртуализации Virtualbox. Была выбрана версия ОС 6.40.9, так как она позволяет реализовать уязвимость “небезопасной конфигурации” на примере файлового сервера SMB.

При создании образа для его работы было выделено 512 Гб ОЗУ, 1 ядро процессора и динамически увеличивающийся виртуальный жесткий диск, максимальной емкостью 2 Гб.

Для обеспечения функционала дополнительных программных и сетевых настроек используется ПО Vagrant. Данное ПО требует дополнительной настройки для обеспечения доступа к базовому образу, но позволяет решить задачу настройки топологии сети.

Дополнительно для обеспечения доступа ПО Vagrant к виртуальной машине, в базовых образах был создан пользователь “Vagrant” с паролем “Vagrant”. Так как такая настройка сама по себе является уязвимостью, позволяющей испытуемому попробовать подключиться к машине со стандартными учетными данными, было принято решение настроить доступ по ключу RSA. Данное решение не позволит испытуемому подключиться к машине без наличия самого ключа, который хранится от него в тайне.

2.6. РАЗРАБОТКА УЯЗВИМЫХ МОДУЛЕЙ

Для того, чтобы соответствовать требованию №5 “В состав ПО должны входить модули, которые позволяют провести проверку навыков, необходимых для поиска наиболее часто встречаемых уязвимостей согласно рейтингу “OWASP TOP 10”. Для начала дадим пояснение, что для данного вида ПО пункт рейтинга №7 “Недостаточная защита от атак” подразумевает под собой отсутствие дополнительных мер по защите инфраструктуры в виде межсетевых экранов (WAF), следовательно выполняется по определению. Пункт №10 “Недостаточное журналирование и мониторинг” также выполняется при отсутствии дополнительных настроек над базовым образом. А пункт №9 “Использование компонентов с известными уязвимостями” подразумевается исходя из целей данного ПО. Следовательно, были реализованы следующие оставшиеся пункты:

Внедрение кода (code injection)

Под внедрением кода понимают возможность злоумышленником внедрить вредоносный код в запрос к серверу приложения таким образом, чтобы сервер выполнил его. Таким образом злоумышленник может получить удаленный доступ к файлам, к которым не должен был получить доступ, и/или непосредственно присоединиться к командной строке посредством reverse shell (обратной оболочки) и выполнения таким образом команд на сервере в обход взаимодействия с приложением.

Для прохождения данного модуля испытуемому необходимо найти страницу приложения, которая посылает запрос к серверу на выполнение команды ping для проверки доступности IP, указанного пользователем. Ввиду некорректной обработки запроса и возможности после непосредственно IP добавить двоеточие и свою команду, появляется возможность просмотра внутренних файлов, недоступных через приложение.

Модуль считается пройденным, если испытуемый прочитал секретный файл и нашел в нем флаг, подтверждающий успешную эксплуатацию уязвимости.

Программный код модуля, в котором содержится данная уязвимость, представлен в приложении №3.

Некорректная аутентификация и управление сессией

Данный вид уязвимости можно подразделить на три вида:

- Перехват сессионных данных - наиболее распространенный вид перехвата осуществляется посредством межсайтового скриптинга, уже упомянутого в рейтинге и реализованного в другом модуле;
- Перебор пароля - данный вид подразумевает, что при отсутствии должных механизмов защиты (например, сброс соединения при большом количестве запросов) и слабой парольной политике (например, если пароли состоят из одних цифр или находятся в списке распространенных паролей) появляется возможность перебрать пароль известного пользователя.

- SQL-инъекция - если аутентификация производится при помощи данных, хранящейся в БД, то, при неправильной настройке и подстановке данных возможна уязвимость SQL-инъекция.

Для данного модуля было решено объединить возможность перебора пароля и SQL-инъекции и предоставить испытуемому несколько вариантов решения. Поэтому при генерации пользователя в базе данных используется один из 100 первых паролей в списке популярных паролей “RockYou”. Запрос к СУБД также не защищен от внедрения SQL-инъекции.

Таким образом, испытуемый должен либо перебрать пароль, используя словарь “RockYou” и автоматизируя процесс, либо использовать SQL-инъекцию и также получить доступ к странице пользователя.

SQL-инъекция представляется возможной ввиду использования некорректного запроса к базе данных, в ходе которого пользовательский ввод никак не контролируется и во вторую переменную можно вставить строку “ ‘ or 1=1; -- -”, которая нарушает логику запроса и отключает дальнейшую проверку пароля пользователя.

Программный код, содержащий данную уязвимость, представлен в приложении №4.

Содержимое файла “docker-compose.yml”, необходимого для запуска инструмента контейнеризации Docker и управления контейнерами PHP-сервера и базы данных MySQL, представлен в приложении №5.

Межсайтовый скриптинг (XSS)

Данная уязвимость подразумевает под собой возможность внедрения вредоносного javascript-кода, который выполняется в браузере пользователя при посещении страницы приложения. Для данной уязвимости был создан модуль, имитирующий работу блога, куда пользователи могут добавить свои новости, уязвимого для данного вида атаки.

Одной из наиболее распространенных целей, для которых используется XSS - кража данных пользователя, например, сессионных данных Cookie.

Поэтому для прохождения данного модуля нужно получить сессионные данные одного из пользователей, действия которого имитируются программно.

Нарушение контроля доступа

Под данной уязвимостью понимают получения злоумышленником доступа к файлам, которые не должны быть ему доступны. Например, доступ к личной странице пользователя приложения.

Для реализации данной уязвимости разработан модуль, имитирующий личный блог пользователя, имеющий как публичные странички, видимые всеми пользователям на главной странице, так и закрытые, доступ к которым осуществляется только если пользователь знает точный адрес URL.

Модуль считается пройденным, если пользователь смог перебрать адреса URL и найти личную закрытую страничку пользователя.

Небезопасная конфигурация

Для реализации данной уязвимости было решено использовать виртуальную машину, содержащий образ RouterOS. В реализации файлового сервера SMB версии 6.40.9, которая была выбрана для создания базового образа сетевого устройства, существует уязвимость CVE 2018-7445. Эксплуатация данной уязвимости возможна при базовых настройках с включенным SMB-сервером.

Для эксплуатации уязвимости испытуемый должен обнаружить данную виртуальную машину, произвести проверку открытых портов и запущенных сервисов, а также узнать версию ОС. Данная информация позволит найти уязвимость и публичный вредоносный код, используемый для эксплуатации уязвимости (эксплойт). После чего найти информацию, подтверждающую прохождение модуля, можно будет в файловом хранилище ОС.

Утечка чувствительных данных

Данная уязвимость может быть схожа с уязвимостью “Нарушения контроля доступа”, но в качестве файла, к которому должен получить доступ испытуемый, выступает конфигурационный файл приложения, содержащий информацию о настройках приложения.

Подделка межсайтовых запросов (CSRF)

Данная уязвимость является развитием XSS уязвимости. Отличием же является то, что сессионные данные пользователя не крадутся, а используются для автоматического посыла на другой сервер для выполнения определенного запроса, например перечисления денег, без ведома пользователя.

Для реализации был сделан сайт, содержащий две страницы:

Первая схожа с описанным в модуле XSS сайтом, куда пользователь может написать свою новость и внедрить таким образом вредоносный код

Второй - непосредственно сайт с тем же оформлением и названием, на котором находится функция раскрытия информации, использовать которую может только администратор.

Действия администратора имитируются программно.

Дополнительным препятствием, которое мешает украсть сессионные данные и подставить их для выполнения запроса является проверка Cookie и IP-адреса, с которого пришел запрос. Таким образом, если Cookie не соответствует IP-адресу, записанному в базе данных при аутентификации - запрос выполнен не будет.

Для прохождения модуля испытуемому необходимо внедрить вредоносный код, выполняемый на стороне пользователя, при посещении страницы администратором будет выполнен код и, соответственно, функция раскрытия информации. После этого, на второй странице будет опубликована информация, при помощи которой испытуемый может подтвердить прохождение модуля.

Для всех модулей испытуемый может удостовериться в правильном нахождении информации, необходимой для доказательства успешной

эксплуатации уязвимости на специальной странице с полем ввода флага для проверки.

2.7. РАЗРАБОТКА АДМИНИСТРАТИВНОЙ ПАНЕЛИ

Для реализации административной панели был выбран свободно распространяемый веб-фреймворк Django, используемый для веб-приложений на языке Python на базе шаблона MVC. В его состав входят технологии ORM, позволяющая связывать объекты с базами данных, системы кеширования и фильтрации данных для защиты от атак, использующих уязвимости внедрения произвольного кода, система аутентификации и авторизации, а также библиотеки для работы с формами.

Были реализованы страницы:

- Аутентификация пользователя - для контроля доступа пользователей к системе, в том числе закрытия несанкционированного доступа пользователей согласно ролевой модели;
- Страница создания и настройки виртуальных машин;
- Страница создания сетевых настроек для объединения виртуальных машин в виртуальную инфраструктуру, а также подключения дополнительных сторонних виртуальных машин;
- Страница перезапуска виртуальных машин;
- Страница проверки полученной информации (“Флага”), подтверждающей успешную эксплуатацию уязвимости;

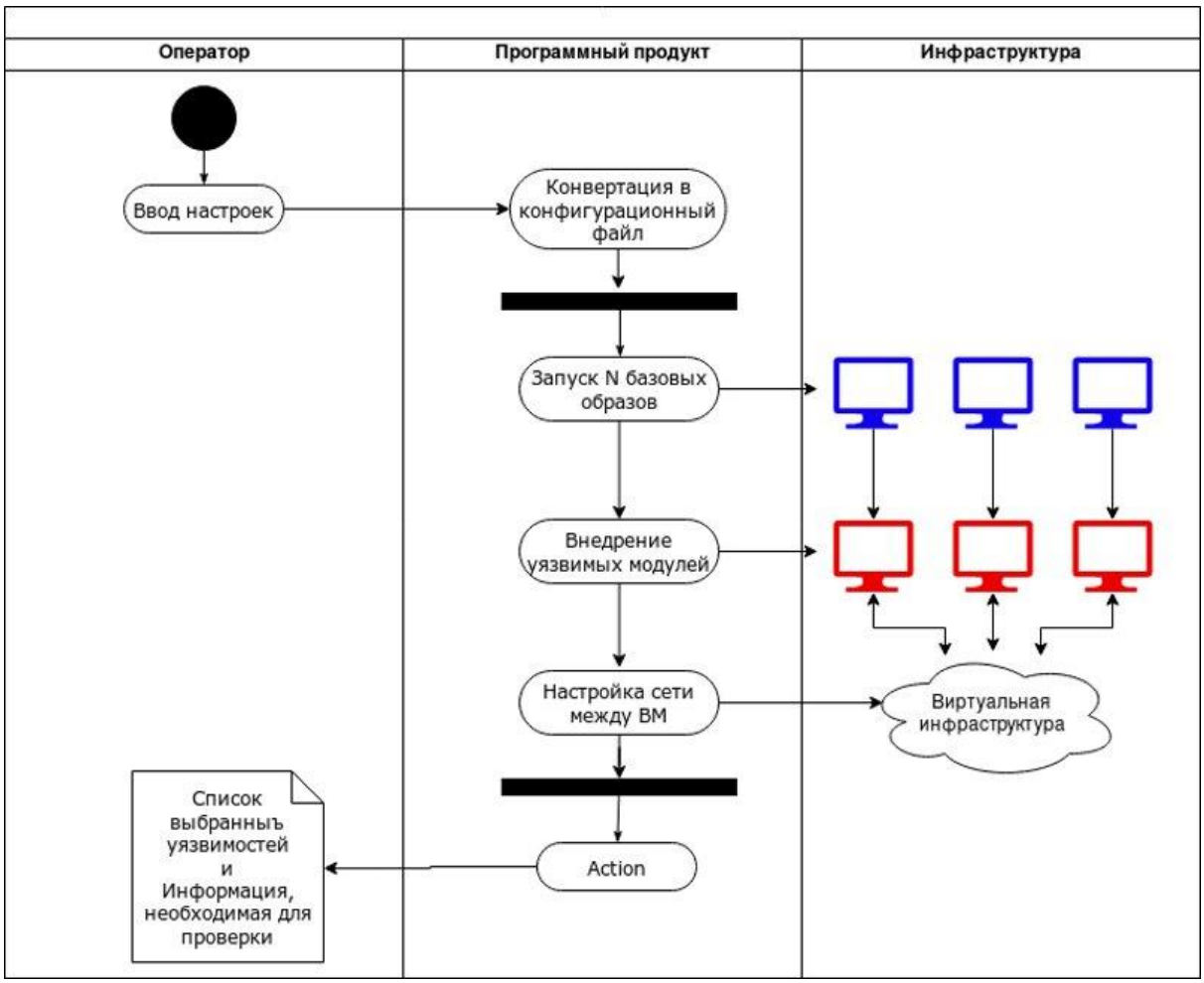

Рис. 6. Работа приложения

После задания настроек виртуальной инфраструктуры, все данные конвертируются в файл для ПО Vagrant, запускаются команды, указанные в уязвимых модулях, необходимые для их внедрения и настройки сети, запускаются виртуальные машины.

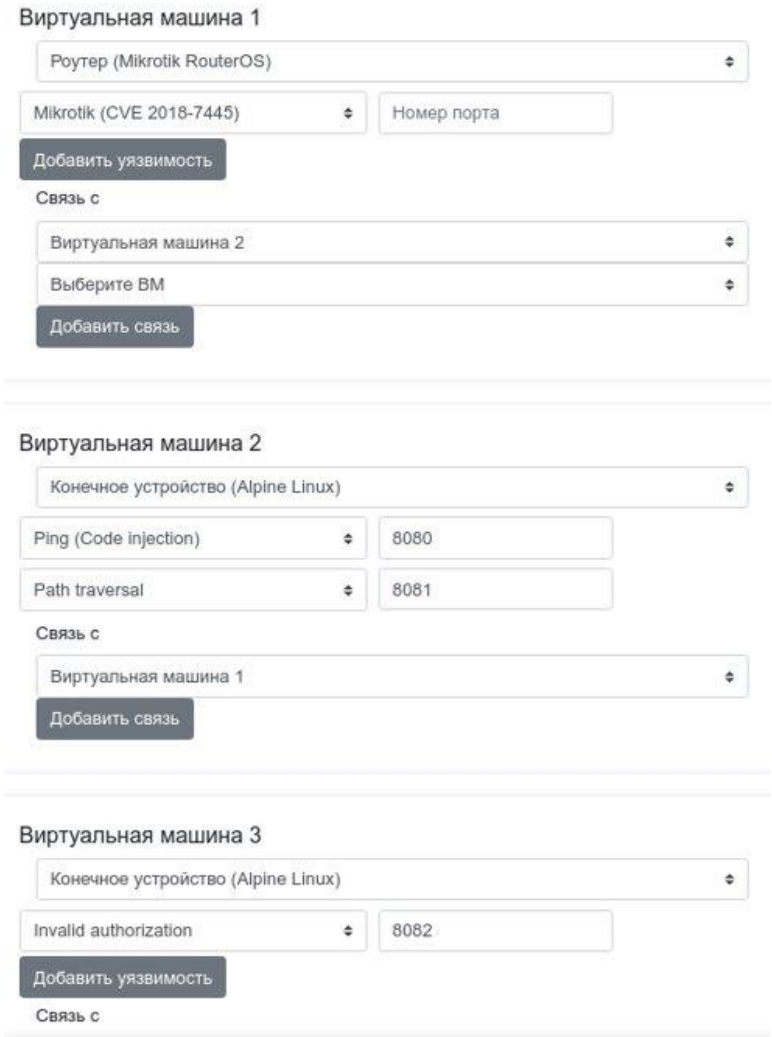

Рис. 7. Интерфейс страницы оператора для создания и настройки УВИ

После этого испытуемый может начать поиск и эксплуатацию уязвимостей, а также проверять нашел ли он нужную информацию на странице проверки флага.

Проверяющий и оператор имеют доступ к специальной странице, на которой можно просмотреть конфигурацию текущей УВИ и всю информацию, необходимую для подтверждения навыка поиска и эксплуатации уязвимостей. Также на данной странице можно остановить работу УВИ.

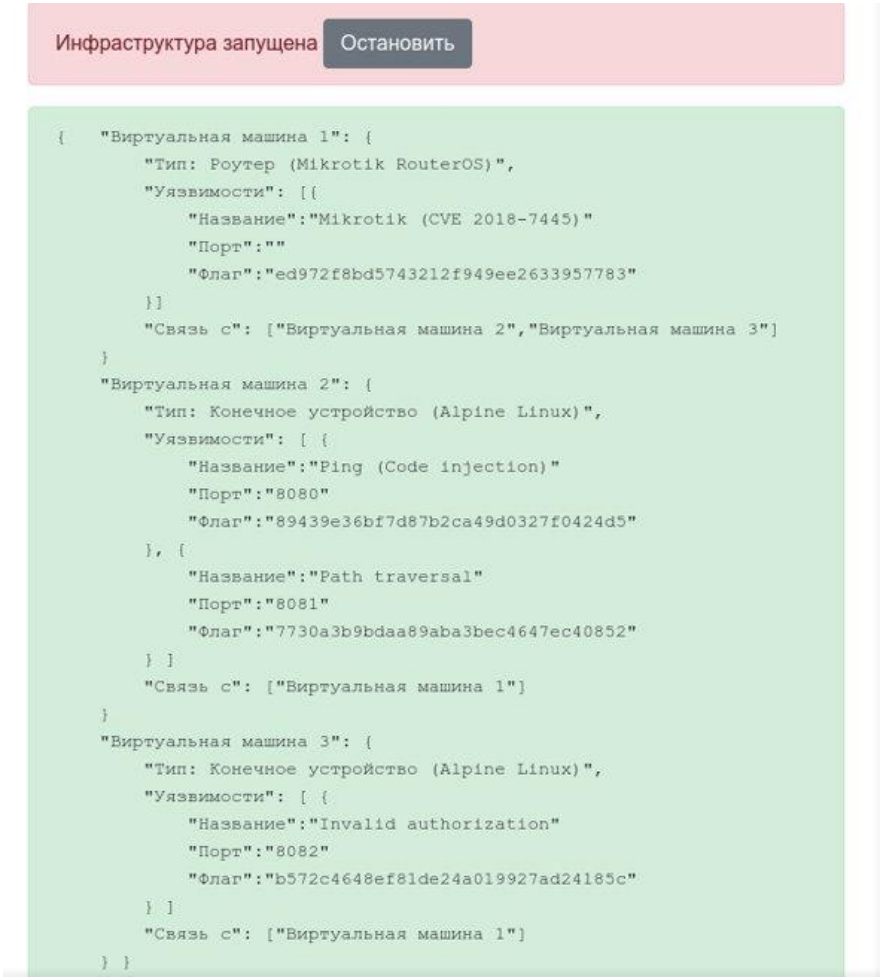

Рис. 8. Страница конфигурации и остановки инфраструктуры

ЗАКЛЮЧЕНИЕ

В рамках данной работы были проанализированы методы проверки практических навыков и умений специалиста по тестированию на проникновение в информационную инфраструктуру, обоснована их необходимость, а также показана актуальность исследований в данной области.

В ходе работы было показано, что существующие методы проверки навыков имеют как положительные, так и отрицательные стороны. Основной выделенной проблемой указанных методов является большая трата времени и ресурсов, которые затрачивают сотрудники при подготовке к проведению оценки испытуемого.

Для решения данной проблемы была поставлена цель: разработать программный продукт для автоматизированного создания уязвимых виртуальных инфраструктур с заданными конфигурациями, использование которых позволит сэкономить время и ресурсы компании, а также объективно оценить практические навыки и умения испытуемого при приеме на работу на должность, связанную с информационной безопасностью.

Для достижения поставленной цели были выделены следующие задачи:

- Исследование особенностей проведения тестирования на проникновение и выделение навыков, которые требуются от специалиста;
- Построение модели уязвимой виртуальной инфраструктуры;
- Исследование существующих разработок в сфере создания виртуальных стендовых лабораторных;
- Исследование особенностей автоматизированного создания виртуальной инфраструктуры и внедрения уязвимостей в них;
- Разработка базового образа операционной системы;
- Разработка базы модулей, содержащих уязвимости, готовых для внедрения в базовый образ;
- Разработка метода внедрения уязвимых модулей в базовый образ;
- Разработка метода создания сети виртуальных машин - виртуальной инфраструктуры;

- Разработка административной панели, которая позволяет производить настройку, запуск и администрирование виртуальной инфраструктуры;
- Тестирование и апробация программного продукта на примере создания наборов инфраструктур с заданными уязвимостями

В ходе решения поставленных задач были разработаны ролевая модель и требования, предъявляемые к ПО на основании ролей, а также уже существующих разработок. На базе данных требований была разработана и реализована архитектура программного продукта для быстрого создания виртуальных машин, внедрения в них разработанных программных модулей и объединение их в виртуальную инфраструктуру, имитирующую реальную, на основе заданных оператором параметров.

Разработанные модули позволяют проверить у испытуемого знания основных популярных уязвимостей, согласно рейтингу “OWASP TOP-10”, а также навыки владения основными инструментами, применяющимися специалистами по тестированию на проникновение на различных этапах модели “Cyber Kill-chain”. Следовательно, виртуальная инфраструктура, получаемая в результате работы программы, может быть использована для объективной оценки навыков специалиста.

Таким образом задачи решены в полном объеме, а цель - достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Abedin M., Nessa S., Al-Shaer E., Khan L. Vulnerability analysis for evaluating quality of protection of security policies // URL: https://dl.acm.org/doi/abs/10.1145/1179494.1179505 (Дата обращения: 08.01.2020)
2. Azam M. H. N., Beuran R. Usability Evaluation of Open Source and Online Capture the Flag Platforms // URL.: https://pdfs.semanticscholar.org/6eea/33700e1b81d82950eaae453950ac75b2cc77.pdf (Дата обращения: 06.01.2020)
3. Chothia T., Novakovic C. An Offline Capture The Flag-Style Virtual Machine and an Assessment of its Value for Cybersecurity Education. // URL: https://research.birmingham.ac.uk/portal/files/21428839/BhamEduVM.pdf (Дата обращения: 15.11.2019)
4. Doupé A., Cova M., Vigna G. Why Johnny can't pentest: An analysis of black-box web vulnerability scanners //International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. – Springer, Berlin, Heidelberg, 2010. – С. 111-131. URL: https://www.academia.edu/19774354/Why_Johnny_Can_t_Pentest_An_Analysis_of_Black-Box_Web_Vulnerability_Scanners
5. Furfaro A., Piccolo A., Parise A., Argento L., Sacca D. A Cloud-based platform for the emulation of complex cybersecurity scenarios // URL: https://sciencedirect.com/science/article/pii/S0167739X1630704X (Дата обращения: 17.12.2019)
6. Furfaro A., Piccolo A., Sacca D., Parise A. A virtual environment for the enactment of realistic cyber security scenarios // URL.: https://ieeexplore.ieee.org/abstract/document/7847720 (Дата обращения: 05.01.2020)
7. Hill J.M.D., Carver C.A., Humphries J.W., Pooch U.W. Using an Isolated Network Laboratory to Teach Advanced Networks and Security // URL:

https://dl.acm.org/doi/abs/10.1145/366413.364533 (Дата обращения: 30.10.2019)

8. Houmb S.H., Franqueria V.N.L., Engum E.A. Quantifying security risk level from CVSS estimates of frequency and impact // URL: https://www.sciencedirect.com/science/article/pii/S0164121209002155 (Дата обращения: 21.12.2019)
9. Hutchins E.M., Cloppert M.J., Amín R.M. lntellígence-Dríven Computer Network Defense informed by Analysis of Adversary Campaigns and intrusion Kill Chains
10. Kapellas A. A thesis in malware development. // URL.: http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/11190/Kapellas_mte1604.pdf?sequence=2 (Дата обращения: 05.01.2019)
11. Papanikolaou A., Vlachos V., Venieris A., Ilioudis C., Papapanagiotou K., Stasinopoulos A. A framework for teaching network security in academic environments // URL: https://www.emerald.com/insight/content/doi/10.1108/IMCS-11-2011-0056/full/html (Дата обращения: 30.11.2019)
12. Schreuders Z.C., Ardern L. Generating randomised virtualised scenarios for ethical hacking and computer security education. // URL: http://z.cliffe.schreuders.org/publications/VibrantWorkshop2015%20-%20Generating%20randomised%20virtualised%20scenarios%20for%20ethical%20hacking%20and%20computer%20security%20education%20%28SecGen%29.pdf (Дата обращения: 15.11.2019)
13. Schreuders Z. C., Shaw T., Muireadhaigh A. M., Staniforth P. Hackerbot: Attacker Chatbots for Randomised and Interactive Security Labs, Using SecGen and oVirt // URL: https://usenix.org/system/files/conference/ase18/ase18_hackerbot.pdf (Дата обращения: 17.12.2019)
14. Schuett M., Rahman S.M. Information Security Synthesis in Online Universities // URL.: https://arxiv.org/abs/1111.1771 (Дата обращения: 06.01.2020)

15. Spring J. M., Hatleback E. Thinking about intrusion kill chains as mechanisms. // URL.: https://academic.oup.com/cybersecurity/article/3/3/185/2804698#110793433 (Дата обращения: 30.11.2019)
16. Szefer J., Lee R. B. Architectural support for hypervisor-secure virtualization //ACM SIGPLAN Notices. – 2012. – Т. 47. – №. 4. – С. 437-450. URL: https://dl.acm.org/doi/abs/10.1145/2248487.2151022 (Дата обращения: 19.02.2019)
17. Tarnowski I. How to use cyber kill chain model to build cybersecurity? // URL.: https://www.eunis.org/download/TNC2017/TNC17-IreneuszTarnowski-cybersecurity.pdf (Дата обращения: 06.01.2019)
18. Tierney S. Knowledge discovery in cyber vulnerability databases. // URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.115.6680&rep=rep1&type=pdf (Дата обращения: 09.12.2019)
19. Wichers D., Williams J. Owasp top-10 2017 // OWASP Foundation. – 2017. URL: https://owasp.org/www-project-top-ten/ (Дата обращения: 09.01.2019)
20. Yadav T., Rao A.M. Technical Aspects of Cyber Kill Chain // URL.: https://link.springer.com/chapter/10.1007/978-3-319-22915-7_40 (Дата обращения: 02.11.2019)
21. Аверченков В.И. Аудит информационной безопасности: учеб. пособие. Брянск: БГТУ, 2005. 269 с.
22. Аверченков В.И., Рытов М.Ю., Кувыклин А.В., Рудановский М.В. Аудит информационной безопасности органов исполнительной власти: учеб. пособие. 3-е изд., стереотип. Москва: ФЛИНТА, 2011. 100 с
23. Алексеев А. Л., Красноперова Е. А., Вахрушева Е. А. Гипервизоры и виртуальные машины // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В НАУКЕ, ПРОМЫШЛЕННОСТИ И ОБРАЗОВАНИИ. – 2019. – С. 121-128.
24. Адрова Л. С., Полежаев П. Н. Сравнительный анализ существующих технологий контейнеризации. – 2016. URL:

http://elib.osu.ru/bitstream/123456789/1955/1/2473-2477.pdf (Дата обращения: 09.03.2019)

25. Артамонова А. А. СРАВНЕНИЕ ГИПЕРВИЗОРНОЙ И КОНТЕЙНЕРНОЙ ТЕХНОЛОГИЙ ВИРТУАЛИЗАЦИИ //Аллея науки. – 2018. – Т. 8. – №. 5. – С. 1146-1152.
26. Бегаев А.Н., Бегаев С.Н., Федотов В.А. Тестирование на проникновение. СПб: Университет ИТМО, 2018. 45 с.
27. Варлатая С.К., Файзенгер А.А., Тимофеева А.И. Определение уязвимостей информационных систем как способ снижения угрозы. // URL: http://izron.ru/articles/aktualnye-voprosy-tekhnicheskikh-nauk-v-sovremennykh-usloviyakh-sbornik-nauchnykh-trudov-po-itogam-m/sektsiya-20-informatsionnye-tekhnologii/opredelenie-uyazvimostey-informatsionnykh-sistem-kak-sposob-snizheniya-ugrozy/ (Дата обращения: 03.12.2019)
28. Вяткин Р. В., Никитина О. И. СРАВНЕНИЕ ВИРТУАЛЬНЫХ МАШИН //Институты и механизмы инновационного развития: мировой опыт и российская практика. – 2017. – С. 230-233. URL: https://www.elibrary.ru/download/elibrary_31756031_58799569.pdf (Дата обращения: 09.03.2019)
29. Горбунов К.С., Семакин А.Е., Зулькарнеев И.Р. Организация внутривузовских соревнований по информационной безопасности в формате CTF. // URL.: http://dspace.kgsu.ru/xmlui/bitstream/handle/123456789/4438/%20%20%20-%20%20_2016_%20%20.pdf?sequence=1#page=12 (Дата обращения: 06.10.2019)
30. ГОСТ Р 51275-2006. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения // URL.: http://docs.cntd.ru/document/1200057516 (Дата обращения: 30.10.2019)
31. ГОСТ Р. 53114-2008. Защита информации. Обеспечение информационной безопасности в организации. Основные термины и определения. – 2009.

URL: http://docs.cntd.ru/document/1200075565 (Дата обращения: 30.10.2019)

32. ГОСТ Р. ИСО/МЭК 27001-2006. Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования //М.: Госстандарт России. – 2008. URL: http://www.gociss.ru/doc/13.GOST_R_ISO_MEK_27001-2006.pdf (Дата обращения: 15.11.2019)
33. Загинайлов Ю. Теория информационной безопасности и методология защиты информации: учебное пособие. Мю-Берлин: Директ-Медиа, 2015. 253 с.
34. Кадан А.М., Доронин А.К. Изучение методов тестирования на проникновение в подготовке специалистов по защите информации. Информатизация образования. 2016
35. Красов А.В., Штеренберг С.И., Москальчук А.И. Методология создания виртуальной лаборатории для тестирования безопасности распределенных информационных систем
36. Марченко А.В., Войналович В.Ю., Воронин С.Н. Анализ состояния системы подготовки специалистов в области информационной безопасности.
37. Меньших В.В., Константиновна Е.К. Оценки доступности информации в телекоммуникационных системах // URL.: https://cyberleninka.ru/article/n/otsenki-dostupnosti-informatsii-v-telekommunikatsionnyh-sistemah (Дата обращения: 05.01.2020)
38. Лазуткин А. Н. Аудит информационной безопасности предприятия. Основные угрозы и этапы внедрения системы обеспечения информационной безопасности // Научно-методический электронный журнал «Концепт», 2016, С. 3211–3215. URL: http://e-koncept.ru/2016/86678.htm (Дата обращения: 30.11.2019)
39. Пантюхин И.С., Дранник А.Л., Птицын А.В. Опыт применения практико-ориентированного подхода к обучению бакалавров основам

информационной безопасности. // URL: https://elibrary.ru/item.asp?id=25030640 (Дата обращения: 30.11.2019)

40. Певнев В. Я. Методы обеспечения целостности информации в инфокоммуникационных системах. // URL: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/vcpitevn_2015_51_16.pdf (Дата обращения: 05.01.2020)
41. Первый рейтинг «Подготовка кадров для ИТ-отрасли в регионах» https://russoft.org/wp-content/uploads/2018/10/Rating_russoft.pdf (Дата обращения: 09.11.2019)
42. Пестунова Т.М., Селифанов В.В., Слонкина И.С., Юракова Я.В. Автоматизированная технология сопоставления угроз и уязвимостей безопасности информации в информационных системах // URL https://repo.ssau.ru/bitstream/Informacionnaya-bezopasnost/Avtomatizirovannaya-tehnologiya-sopostavleniya-ugroz-i-uyazvimostei-bezopasnosti-informacii-v-informacionnyh-sistemah-64901/1/156-160.pdf (Дата обращения: 09.12.2019)
43. Плешков А. С., Рудер Д.Д. Тестирование на проникновение как анализ защищенности компьютерных систем // URL: https://cyberleninka.ru/article/n/testirovanie-na-proniknovenie-kak-analiz-zaschischennosti-kompyuternyh-sistem (Дата обращения: 09.12.2019)
44. Положение Банка России от 09.06.2012 N 382-П (ред. от 07.05.2018) "О требованиях к обеспечению защиты информации при осуществлении переводов денежных средств и о порядке осуществления Банком России контроля за соблюдением требований к обеспечению защиты информации при осуществлении переводов денежных средств" // URL: http://www.consultant.ru/document/cons_doc_LAW_131473/ (Дата обращения: 10.11.2019)

45. СЗ Р. Ф. Уголовный кодекс Российской Федерации от 13.06. 1996 № 63-ФЗ. – 1996. (ред. от 07.04.2020)
46. Соловьев В.А., Шемяхина А.К. Разведка в открытых источниках как первый этап кибератаки в модели Cyber Kill Chain // URL.: https://elibrary.ru/item.asp?id=25327504 (Дата обращения: 06.01.2019)
47. Статистический сборник “Состояние преступности в России за июль 2019 г.”. URL: https://www.genproc.gov.ru/upload/iblock/9fb/sbornik_7_2019.pdf (дата обращения: 30.11.2019)
48. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. СПб.: Питер, 2012. 960 с. (Дата обращения: 03.12.2019)
49. Урбанович П. П., Романенко Д. М, Кабак Е. В. Компьютерные сети: учеб. пособие для студентов высших учебных заведений по техническим специальностям. Минск: БГТУ, 2011. 400 с. https://elib.belstu.by/bitstream/123456789/3084/1/urbanovich_kompyuternye-seti.2011.pdf (Дата обращения: 15.11.2019)
50. Федорченко А.В., Чечулин А.А., Котенко И.В. Исследование открытых баз уязвимостей и оценка возможности их применения в системах анализа защищенности компьютерных сетей. // URL: https://cyberleninka.ru/article/n/issledovanie-otkrytyh-baz-uyazvimostey-i-otsenka-vozmozhnosti-ih-primeneniya-v-sistemah-analiza-zaschischennosti-kompyuternyh-setey (Дата обращения: 21.12.2019)
51. Цуканова О.А., Смирнов С.Б. Экономика защиты информации: Учебное пособие. СПб.: СПб ГУИТМО, 2013. 59 с.
52. Шафигуллина А. Р. ПЛАТФОРМА DOCKER: КОНТЕЙНЕРНАЯ ВИРТУАЛИЗАЦИЯ //НАУЧНОЕ СООБЩЕСТВО СТУДЕНТОВ. МЕЖДИСЦИПЛИНАРНЫЕ ИССЛЕДОВАНИЯ. – 2019. – С. 51-55. URL: https://www.elibrary.ru/item.asp?id=40356425 (Дата обращения: 10.02.2019)

Программный код объектов модели

```
import os
import binascii
import json
import fileinput
import paramiko
import pyvbox

class base_vm:
    interfaces = []
    vulnerabilities = []

    def __init__(self, vm_os, ip, name):
        self.name = name
        self.IP = ip
        self.OS = vm_os

    def start(self):
        vbox = virtualbox.VirtualBox()
        session = virtualbox.Session()
        machine = vbox.find_machine(machine_name)
        proc = machine.launch_vm_process(session, "headless")
        proc.wait_for_completion(timeout=-1)

    def stop(self):
        vbox = virtualbox.VirtualBox()
        session = virtualbox.Session()
        vm = vbox.find_machine(self.name)
        session.console.power_down()
```

```
    def connect(self, ip):
        vbox = virtualbox.VirtualBox()
        session = virtualbox.Session()
        vm = vbox.find_machine(self.name)
        session = vm.create_session()
        gs = session.console.guest.create_session('vagrant', 'vagrant')

        while True:
            command = raw_input('$ ')
            if command == 'exit':
                break
            while True:
                process, stdout, stderr = gs.execute(command)
                print stdout

    def execute_command(self, command):
        vbox = virtualbox.VirtualBox()
        vm = vbox.find_machine(self.name)
        session = vm.create_session()
        gs = session.console.guest.create_session('vagrant', 'vagrant')
        process, stdout, stderr = gs.execute(command)
        print stdout



class vulnerability:
    def __init__(self, settings_file):
        with open(settings_file) as json_file:
            data = json.load(json_file)
            for p in data['vulnerability']:
```

```
            name = p['name']
            desc = p['description']
            type_owasp = p['type_owasp']
            software = p['soft']
            port = p['port']
            docker_usage = p['docker_usage']

    def generate_flag(self, folder):
        flag = binascii.b2a_hex(os.urandom(15))
        template = "^FLAG^"
        flag_changed = False
        for root, dirs, files in os.walk(folder):
            for file in files:
                with fileinput.FileInput(current_file, inplace=True, backup='.bak') as file:
                    for line in file:
                        if template in line:
                            print(line.replace(template, flag), end='')
                            flag_changed = True
            if flag_changed:
                break



    def integrate(self):
        sshClient = paramiko.SSHClient()
        sshClient.connect(ip, username='vagrant', password='vagrant')
        channel = sshClient.get_transport().open_session()
        channel.get_pty()
        channel.invoke_shell()

        while True:
```

```
            command = raw_input('$ ')
            if command == 'exit':
                break
            channel.send(command + '\n')
            while True:
                if channel.recv_ready():
                    output = channel.recv(1024)
                    print output
                else:
                    time.sleep(0.5)
                    if not(channel.recv_ready()):
                        break
        sshClient.close()


class base_net_vm(base_vm):
    routes = []

class routes:
    def __init__(self, ip_from, ip_to):
        from = ip_from
        to = ip_to
```

Приложение 2

Содержимое шаблона файла конфигурации

```
out_net = "^IP_EXTERNAL^"
in_net = "^IP_INTERNAL^"
vms = [
  ^VM_CONFIGS_HERE^
  # Пример настроек конфигурации виртуальных машин
  #
  #  {
  #    :hostname => "vm1.",
  #    :base_vm => "^BASE_VM_OS^",
  #    :ip => out_net + "150",
  #    :ip_int => in_net + "1",
  #    :config_file = "/src/shared_folder/^MODULE_NAME^/start.sh",
  #    :ram => 2000
  #  }
]

Vagrant.configure(2) do |config|
  config.vm.synced_folder "/home/user/soft/modules", "/src/shared_folder"
  config.vm.synced_folder ".", "/vagrant", disabled: true

  servers.each do |machine|
    config.vm.define machine[:hostname] do |node|
      node.vm.box = "base_vm"
      node.vm.usable_port_range = (0..65535)
      node.vm.hostname = machine[:hostname]
      if (!machine[:ip].nil?)
```

```
            node.vm.network "public_network", ip: machine[:ip], bridge: 'Intel(R)
Centrino(R) Wireless-N 130'
          end
          if (!machine[:ip_int].nil?)
            node.vm.network "private_network", ip: machine[:ip_int],
virtualbox__intnet: "intnet"
          end
          node.ssh.host = machine[:ip]
          node.ssh.private_key_path = "/home/user/soft/private_key"
          node.ssh.username = "vagrant"
          node.vm.provider "virtualbox" do |vb|
            vb.customize ["modifyvm", :id, "--memory", machine[:ram]]
            vb.name = machine[:hostname]
            if (!machine[:hdd_name].nil?)
              unless File.exist?(machine[:hdd_name])
                vb.customize ["createhd", "--filename", machine[:hdd_name], "--size",
machine[:hdd_size]]
              end
              vb.customize ["storageattach", :id, "--storagectl", "SATA", "--port", 1, "-
-device", 0, "--type", "hdd", "--medium", machine[:hdd_name]]
            end
          end
          if (!machine[:config_file].nil?)
            config.vm.provision :shell, path: machine[:config_file]
          end
        end
    end
end
```

Программный код, содержащий уязвимость “Внедрение кода”

```
<html>
<head>
   <title>Ping test</title>
</head>
<body>
   <form method="GET" action="">
     <input type="text" name="host" />
     <input type="submit" value="ping host" />
   </form>
   <?php
      if(isses($_post['submit'])){
         $target = $_REQUEST['ip'];

         if(stristr(php_uname('s'), 'Windows NT')){
            $cmd = shell_exec('ping ' . $target);
         }
         else {
            $cmd = shell_exec('ping -c 4 '. $target);
         }
         echo "<pre>{$cmd}<pre>";
   }
   ?>
</body>
</html>
```

Программный код, содержащий уязвимость некорректной аутентификации

```
<html>
<head>
   <title>Authorization</title>
</head>
<body>
   <form method="GET" action="">
   <p>Username: <input type="text" name="uname" /></p>
   <p>Password: <input type="text" name="password" /></p>
   <p><input type="submit" /></p>
   </form>

   <?php
      if (isset($_POST['uname'])){
         if (isset($_POST['password'])){
            $name = $_POST['uname'];
            $pass = $_POST['password'];

            $mysqli = new mysqli('localhost', 'dbuser', 's3cur3_p@55',
'incorrect_auth');

            if ($mysqli->connect_errno) {
               printf("Connect failed: %s\n", $mysqli->connect_error);
               exit();
            }
            $sql = "SELECT username FROM users WHERE username = '" . $name .
"' AND password = '" . $pass . "';";
            if ($result = $mysqli->query($sql)) {
```

```
                while($obj = $result->fetch_object()){
                    $localIP = getHostByName(getHostName());
                    header("Location: {$localIP}/wellDone.php");
                    die();
                }
                }
                elseif($mysqli->error){
                    print($mysqli->error);
                    echo '<h3 style="color:red">Incorrect password</h3>';
                }
            }
            else {
                echo '<h3 style="color:red">Incorrect password</h3>';
            }
        }
        else{
            echo '<h3 style="color:red">Incorrect login</h3>';
        }
    ?>
</body>
</html>
```

Приложение 5

Содержимое файла “docker-compose.yml” модуля с уязвимостью некорректной аутентификации

```
version: '3'

services:
  db:
    image: mysql:5.7
    command: CREATE TABLE users (username text, password text)
    command: INSERT INTO users (username, password) VALUES ("admin",
"^PASSWORD^")
    environment:
      MYSQL_ROOT_PASSWORD: 5tr@n63
      MYSQL_DATABASE: incorrect_auth
      MYSQL_USER: dbuser
      MYSQL_PASSWORD: s3cur3_p@55
    ports:
      - "3306:3306"
  web:
    image: php:7.2.2-apache
    container_name: php_web
    depends_on:
      - db
    volumes:
      - ./php/:/var/www/html/
    ports:
      - "^PORT_2^:80"
    stdin_open: true
    tty: true
```