

**«Санкт-Петербургский государственный электротехнический университет  
«ЛЭТИ» им. В.И.Ульянова (Ленина)»  
(СПбГЭТУ «ЛЭТИ»)**

<b>Направление</b>	09.04.04 Программная инженерия
<b>Программа</b>	Автономные интеллектуальные системы
<b>Факультет</b>	КТИ
<b>Кафедра</b>	МОЭВМ

*К защите допустить*

Зав. кафедрой

А.А. Лисс

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
МАГИСТРА**

**Тема: ДИНАМИЧЕСКОЕ АВТОМАТИЗИРОВАННОЕ  
СКАНИРОВАНИЕ СЕТЕВОГО ПЕРИМЕТРА ДЛЯ ОБНАРУЖЕНИЯ  
УЯЗВИМОСТЕЙ С ПОМОЩЬЮ OPENSOURCE ИНСТРУМЕНТОВ**

Студент		<hr/>	Д.С. Занин
		<i>Подпись</i>	
Руководитель	к.т.н., доцент <i>(Уч. степень, уч. звание)</i>	<hr/>	А.А. Лисс
		<i>Подпись</i>	
Консультанты		<hr/>	И.А. Юшкевич
	<i>(Уч. степень, уч. звание)</i>	<i>Подпись</i>	
	к.т.н., доцент <i>(Уч. степень, уч. звание)</i>	<hr/>	М.Н. Магомедов
		<i>Подпись</i>	
	к.т.н., доцент <i>(Уч. степень, уч. звание)</i>	<hr/>	М.М. Заславский
		<i>Подпись</i>	

Санкт-Петербург

2024

## ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю  
Зав. кафедрой МОЭВМ  
\_\_\_\_\_ А.А. Лисс  
«\_\_\_» \_\_\_\_\_ 20\_\_ г.

Студент      Занин Д.С.      Группа 8306

Тема работы: Динамическое автоматизированное сканирование сетевого периметра для обнаружения уязвимостей с помощью OpenSource инструментов

Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» кафедра МОЭВМ

Исходные данные (технические требования): Поставлена задача разработки приложения осуществляющего динамическое автоматизированное сканирование периметра сети OpenSource инструментами с возможностью их добавления. Реализовать графический интерфейс на Django. Обеспечить кроссплатформенность обернув приложение и скрипт в Docker контейнер.

Содержание ВКР: Введение, Современное состояние вопроса, Описание разработки, Результаты экспериментов, Заключение.

Перечень отчетных материалов: пояснительная записка, иллюстративный материал

Дополнительные разделы: Оценка и защита результатов интеллектуальной деятельности

Дата выдачи задания	Дата представления ВКР к защите
«___» _____ 20__ г.	«___» _____ 20__ г.

Студент	_____	Д.С. Занин
---------	-------	------------

Руководитель	к.т.н., доцент (Уч. степень, уч. звание)	_____	А.А. Лисс
--------------	---	-------	-----------

Консультант	_____	И.А. Юшкевич
-------------	-------	--------------

(Уч. степень, уч. звание)

# КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю  
Зав. кафедрой МОЭВМ  
\_\_\_\_\_ А.А. Лисс  
«\_\_» \_\_\_\_\_ 20\_\_ г.

Студент      Занин Д.С.      Группа 8306  
Тема работы: Динамическое автоматизированное сканирование сетевого периметра для обнаружения уязвимостей с помощью OpenSource инструментов.

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме	05.02 – 05.03
2	Обзор и анализ OpenSource инструментов.	06.03 – 12.03
3	Разработка скриптов, создание web-приложения, оборачивание в docker контейнер.	13.03 – 25.04
4	Проведение тестирования и доработка приложения.	26.04 – 29.04
5	Оформление пояснительной записки	30.04 – 04.05
6	Оформление демонстрационного материала, подготовка доклада	04.05 – 06.05
7	Предзащита	10.05.2024

Студент \_\_\_\_\_ Д.С. Занин  
*Подпись*

Руководитель      к.т.н., доцент  
(Уч. степень, уч. звание) \_\_\_\_\_ А.А. Лисс  
*Подпись*

Консультанты \_\_\_\_\_ И.А. Юшкевич  
(Уч. степень, уч. звание)      *Подпись*

## РЕФЕРАТ

Пояснительная записка 71 стр., 16 рис., 9 табл., 36 ист.

OPENSOURCE ИНСТРУМЕНТЫ, WORKFLOW, WEB-ПРИЛОЖЕНИЕ, DOCKER, DJANGO, NMAP, AMASS, FFUF, NUCLEI, SUBFINDER.

**Объектом исследования** создание возможности объединения OpenSource инструментов в общий рабочий процесс и возможность их добавления, а также возможности объединения работы различных инструментов для сканирования периметра.

**Предметом исследования** Автоматизация, оценка и исследование возможности доступных инструментов для тестирования на проникновение.

**Цель работы:** Основная задача состоит в написании программного продукта, который позволяет автоматизировать сканирование сетевого периметра с использованием известных инструментов.

Сканирование периметра сети организации на наличие уязвимостей должно помочь организации найти изъяны в её сети и не позволить злоумышленнику воспользоваться ими. В данной работе освещаются результаты, полученные по этой теме. Производится демонстрация графического интерфейса. Представлена работа тех OpenSource инструментов, которые рассматривались в обзоре литературы и использовались в данной работе для решения поставленной проблемы, также применялись и другие инструменты в процессе создания итогового приложения.

## **ABSTRACT**

Scanning the perimeter of an organization's network for vulnerabilities should help the organization find flaws in its network and prevent an attacker from taking advantage of them. This paper highlights the results obtained on this topic.

The graphical interface is being demonstrated.

The work of those OpenSource tools that were considered in the literature review and used in this work to solve the problem is presented, and other tools were also used in the process of creating the final application.

## СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ .....	8
ВВЕДЕНИЕ .....	9
1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ.....	11
1.1 Краткая сводка статистики защищённости сетевого периметра компаний .....	11
1.2 Статистика ущерба от киберприступности .....	12
1.3 Краткая сводка выводов полученных в данном обзоре .....	14
2 ОПИСАНИЕ РАЗРАБОТКИ .....	16
2.1 Добавление и обзор OpenSource инструментов. ....	16
2.1.1 Subfinder.....	16
2.1.2 ffuf .....	18
2.1.3 Nmap.....	19
2.1.4 Nuclei.....	20
2.2 Сохранение полученных результатов в таблицы базы данных.....	22
2.3 Замена Фреймворка flask на Django .....	24
2.3.1 Изменение в работе сервера .....	26
2.3.2 Изменение в работе скрипта.....	26
2.4 Использование Docker .....	29
2.5. Обзор аналогов Open Source инструментов, используемых в работе, но не вошедших в данную работу .....	30
2.5.1 Amass .....	34
2.5.2 Assetfinder .....	38
2.5.3 Dirsearch.....	40

3 РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ .....	41
3.1 Полученные результаты по скорости работы приложения .....	41
3.2 Анализ полученных результатов .....	41
3.3 Краткие выводы по полученным результатам .....	46
3.4 Расположение результатов работы .....	47
4 ОЦЕНКА И ЗАЩИТА РЕЗУЛЬТАТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ.....	48
4.1 Описание результата интеллектуальной деятельности .....	48
4.2 Оценка рыночной стоимости результата интеллектуальной деятельности .....	52
4.3 Правовая защита результатов интеллектуальной деятельности .....	60
4.3.1 Время в течение которого авторские права будут в силе .....	60
ЗАКЛЮЧЕНИЕ.....	66
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	68

## **ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

ЯП – язык программирования;

ОС – операционные системы;

ПО – программное обеспечение;

ВКР – выпускная квалификационная работа;

БД – база данных;

ЮЛ – юридические лица.

ГК – гражданский кодекс

РФ - Российская Федерация

ЭВМ – электронно вычислительные машины

ВКР - Выпускная квалификационная работа

ЗП - заработная плата



## ВВЕДЕНИЕ

В современном цифровом мире, где безопасность в сети играет ключевую роль, создание эффективных инструментов для обнаружения уязвимостей на веб ресурсах является неотъемлемой частью стратегии обеспечения информационной безопасности.

Данная работа посвящена созданию возможности объединения инструментов в единый workflow и возможность добавления новых OpenSource инструментов, цель которого – проведение сканирования периметра сети с использованием OpenSource инструментов.

Также в данной работе реализована технология Docker контейнеризации для обеспечения изолированности и кроссплатформенности приложения.

**Актуальность** — Сканирование сетевого периметра на наличие уязвимостей - задача актуальная, и довольно часто требующая своего регулярного выполнения. Однако современные готовые продукты создаваемые для решения поставленной задачи, предлагаемые на рынке, имеют ряд недостатков. А именно, подобные решения, зачастую могут разрабатываться для конкретной компании или задачи, трудно встраиваются и адаптируются под необходимые требования, в недостатки также можно выделить высокую стоимость, и что часто бывает – медленную скорость работы. Зачастую требуется не доскональная проверка сетевого периметра на наличие всех возможных уязвимостей, а быстрая проверка на наличие критических уязвимостей в периметре сети. Терминальные OpenSource инструменты достаточно гибкие и универсальные, а графический интерфейс управляющий ими делает их использование понятным и простым. Совокупность всех перечисленных проблем, делает актуальным создание программы способной их решить.

**Объектом исследования** создание возможности объединения OpenSource инструментов в общий рабочий процесс и возможность их добавления, а также возможности объединения работы различных инструментов для сканирования периметра.

**Предметом исследования** Автоматизация, оценка и исследование возможности доступных инструментов для тестирования на проникновение.

**Цель работы:** Основная задача состоит в написании программного продукта, который позволяет автоматизировать сканирование сетевого периметра с использованием известных инструментов.

**Практическая значимость** исследования заключается в создании инструмента для обнаружения уязвимостей в периметре сети цели, которое позволит автоматизировать сканирование сетевого периметра с использованием известных OpenSource инструментов. Технология Docker контейнеризации позволит быть данному инструменту изолированным и кроссплатформенным, а применение фреймворка Django позволит данному приложению быть переносимым в другие проекты при необходимости. Данный подход позволит использоваться полученный инструмент на любой ОС и в любом web-проекте, написанном на Django. Полученный инструмент просканировав цель сможет дать пользователю представление о списке поддоменов целевого домена, ip-адресах хостов на которых расположены эти домены, найти открытые порты на найденных хостах и версии ОС и ПО, а также сервисы расположенные на данных портах. Данный инструмент соберёт информацию о скрытых файлах и каталогах на сайте цели, атак же найдёт список уязвимостей на найденных открытых портах. Все найденные данные записываются в БД. Полученный инструмент можно применять для поиска уязвимостей на целевом web ресурсе и получения представления о периметре сети цели.

# **1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Краткая сводка статистики защищённости сетевого периметра компаний**

По результатам Positive Technologies, исследования инструментального анализа защищенности сетевых периметров корпоративных информационных систем [1] была получена печальная статистика:

1. На внешних сетевых ресурсах 84% организаций выявлены уязвимости высокого уровня риска. То есть, другими словами, 84% изученных компаний практически не защищены. [2]

2. Каждой десятой уязвимостью может воспользоваться низкоквалифицированный злоумышленник. [2]

3. Каждая вторая уязвимость может быть устранена установкой актуальных обновлений ПО. [2]

4. В 74% организаций служба SSH напрямую доступна для подключения из интернета. В то же время каждая пятая уязвимость в программном обеспечении связана с ошибками в OpenSSH, которые могут привести к получению контроля над ресурсами сетевого периметра или проникновению в локальную сеть. [2]

5. На периметре всех организаций выявлены узлы, уязвимые для атаки SWEET32, а для 84% из них до сих пор актуальна атака POODLE. Если злоумышленнику удастся реализовать эти атаки, это даст возможность извлечь конфиденциальные данные из зашифрованных соединений. [2]

6. Для 10% выявленных уязвимостей существуют общедоступные эксплойты, а значит — каждую десятую уязвимость злоумышленник может поэксплуатировать, даже не имея профессиональных навыков программирования или опыта обратной разработки. [2]

7. В 84% организаций выявлены уязвимости высокого уровня риска. В 58% организаций обнаружены уязвимости высокого уровня риска, для которых существуют общедоступные эксплойты. [2]

## **1.2 Статистика ущерба от киберприступности**

В последние годы мир столкнулся с резким увеличением киберпреступлений, что привело к значительным финансовым потерям и угрозам для информационной безопасности. Эксперты отмечают тенденцию к увеличению частоты и сложности кибератак, что требует усиленных мер по защите цифровой инфраструктуры. Статистические данные свидетельствуют о значительном росте числа атак на информационные системы, что подчёркивает необходимость повышения уровня кибербезопасности как в частном, так и в государственном секторах.

Существенное увеличение ущерба от киберпреступлений за последние годы указывает на то, что традиционные методы защиты уже не способны справиться с новыми угрозами. Это вынуждает организации пересматривать свои подходы к обеспечению информационной безопасности и инвестировать в более современные и эффективные решения. Отчёты о росте числа утечек данных и мошенничества с учётными записями подтверждают, что киберприступность становится всё более изощрённой и разнообразной.

В свете этих данных, становится очевидной необходимость комплексного подхода к защите от киберугроз, который включает в себя не только технологические инновации, но и образовательные программы, направленные на повышение осведомлённости о кибербезопасности.

Статистика ущерба от киберприступности.

1. Общий ущерб от киберприступности оценивается в пределах 90 миллиардов рублей. Для сравнения, в 2020 году ущерб от киберпреступлений оценивался в районе 70 миллиардов рублей [3].

2. По оценкам международных экспертных сообществ, ущерб, наносимый кибератаками, будет ежегодно увеличиваться, и в 2025 году ущерб от киберпреступлений достигнет 10,5 трлн долларов ежегодно [3]

3. В I квартале 2022 года количество атак увеличилось на 14,8% по сравнению с IV кварталом 2021 года. [4] Во второй половине квартала

проводились множественные атаки на веб-ресурсы, и их доля выросла до 22% по сравнению с 13%, наблюдаемыми в прошлом квартале [4]

4. ФБР сообщило, что после пандемии COVID-19 количество сообщений о киберпреступлениях увеличилось на 300% [5]

5. Ожидается, что к 2022 году глобальные расходы на кибербезопасность достигнут 170,4 миллиарда долларов. [5]

6. 69% компаний не считают, что угрозы, которые они видят, могут быть блокированы установленными антивирусными программами. [5]

7. Статистика 2021 - 2022 года показывает. Мошенничество с похищением учетных записей выросло на 850% [6]

8. В 2020 году предполагаемый ущерб от киберпреступлений за два года увеличился более чем на 50% и превысил 1 трлн долларов (The Hidden Costs of Cybercrime, McAfee) [7].

9. Средняя стоимость утечки данных выросла до 4,24 млн долларов. Это самый высокий показатель за последний 17 лет [7].

10. Количество утечек данных в сфере здравоохранения увеличилось на 55% в 2020 году по сравнению с предыдущим годом (Healthcare Breach Report 2021, Bitglass) [7].

11. Статистика за 2022 год: только с 1 по 5 марта количество атак на российские IT-системы утроилось по сравнению с февралем 2021 года. [8].

12. Вместо бурного развития рынка ограничения и снижение конкуренции привели к бурному росту цен. Хотя около 70% российских решений уступают по функционалу до зарубежных [8].

13. Затраты на реализацию федерального проекта «Информационная безопасность» нацпрограммы «Цифровая экономика» составят 35 млрд руб. за период до 2024 г. [9]

14. Текущая редакция документа предполагает расходы за период до 2024 г в размере 34,6 млрд руб. Большая часть этой суммы будет выделена федеральным бюджетом — 33,5 млрд руб. [9]

Итоги статистики, не защищёны даже тех компаний, на безопасность которых глобальные расходы в общей сложности составляют 170,4 миллиарда. А предприятия, не заботящиеся о своей защите от киберугроз беззащитны перед злоумышленниками.

На решение данной проблемы выделяется десятки миллиардов рублей, и сфера информационной безопасности активно развивается. Всё равно рост ущерба и числа кибератак растёт. В некоторых направлениях сферы IT число атак и ущерб от них растёт экспоненциально.

Цены на готовые решения в области информационной безопасности растут и некоторые решения, из-за их цен, не доступны для малого и среднего бизнеса, а также частным лицам. Качество этих решений зачастую не соответствует требуемому уровню.

Сетевой периметр отгораживает предприятие от всех внешних воздействий, от всего глобального интернета в целом. Статистика показывает, что предприятия, которые не заботятся о безопасности сети своего предприятия могут быть не защищены. Из всего перечисленного можно сделать вывод, что в сфере IT, решаемая задача является по-настоящему актуальной и важной, как для вопроса финансового характера, так и непосредственно информационного.

### **1.3 Краткая сводка выводов полученных в данном обзоре**

Незащищённость сетевого периметра компаний это частое явление, даже в тех компаниях где выделяются средства на обеспечение информационной безопасности периметра сети.

Даже при том, что на развитие информационной безопасности и на решение проблем связанных с ней выделяются десятки миллиардов рублей количество атак и ущерб от них растёт с каждым годом.

Компании, которые не заботятся о защите своих web-приложений и периметра сети в, которых эти приложения работают, не защищены. Часто подобный сетевой периметр может взломать не профессиональный пен тестер.

Причиной этого часто является открытые порты сети или не удалённые копии баз данных, например, с логинами и паролями пользователей.

Цены на готовые решения в области информационной безопасности растут и в некоторых случаях не доступны для малого и среднего бизнеса, а также для частных лиц.

## **2 ОПИСАНИЕ РАЗРАБОТКИ**

### **2.1 Добавление и обзор OpenSource инструментов.**

#### **2.1.1 Subfinder**

Является инструментом для обнаружения субдоменов. Он также использует пассивные методы. Он может, как и OpenSource инструмент Amass собирать информацию о субдоменах из онлайн источников, использует интернет архивы, поисковые двигатели. Информация на стороннем источнике о способности Subfinder к работе с собранной информацией о поддоменах (найденные данные используется в модуле пермутации, вдохновлённом altdns, для генерации изменённых имён и быстрой их проверки используя мощный движок брут-форса. Ещё, умеет выполнять обычный брут-форс.) [10]

Так как данный инструмент использует пассивный способ поиска, то он работает довольно быстро, а также его можно без опасений использовать как инструмент тестирования на проникновение. И как сказано на официальной странице GitHub - это инструмент обнаружения поддоменов, который возвращает допустимые поддомены для веб-сайтов, используя пассивные онлайн-источники. Он имеет простую модульную архитектуру и оптимизирован для повышения скорости. [11]

Также на официальной страницы прямо указана цель, для которой создавался этот инструмент: subfinder создан только для выполнения одной задачи - пассивного перечисления поддоменов, и он делает это очень хорошо. [11]

Subfinder поддаётся интеграции с другими инструментами. Это значит, что если Subfinder собрал информацию о поддоменах, то другой инструмент сможет использовать эту информацию.

5 основных плюсов, которыми обладает этот инструмент:

1. Быстрые и мощные модули разрешения и устранения подстановочных знаков [11].



2. Тщательно подобранные пассивные источники для достижения максимальных результатов [11].
3. Поддерживается несколько выходных форматов (JSON, file, stdout) [11]
4. экономичный расход ресурсов [11]
5. STDIN/OUT поддержка стандартного ввода-вывода обеспечивает легкую интеграцию в рабочие процессы [11]

В сравнении с другими подобными OpenSource инструментами именно subfinder оказался наиболее быстрым и простым в использовании.

Так, например аналогичный инструмент Amass собрал информацию о поддоменах за 3 минуты 58 секунд, а subfinder справился за 7 секунд.

По результатам исследования было принято решение использовать subfinder для сбора поддоменов целевого домена. Он показал своё преимущество в скорости, а остальные OpenSource инструменты имели ряд недостатков, среди которых не стабильная работа, медленный процесс сканирования цели. Было принято решение, что прочие подобные инструменты влияют на производительность в худшую сторону и увеличивают затрачиваемое время на сбор поддоменов из-за сложности настройки, блокировки по IP и зависимости от сторонних сервисов.

Поэтому в данной работе использовался именно subfinder, который хорошо себя зарекомендовал. Он позволил обеспечить надёжность и скорость в процессе автоматизированного, динамического сканирования целевого периметра сети, а также subfinder позволил эффективнее использовать ресурсы. Прирост в скорости, при переходе на subfinder увеличился в 34 раза по сравнению с аналогичными OpenSource инструментами используемые в разработке прошлых семестров. Повышение скорости работы по сбору поддоменов увеличит и скорость всего процесса сканирования цели.

### 2.1.2 ffuf

Это инструмент играет роль фазера. На официальной странице GitHub, указанной в списке источников под номером 11, указана ссылка на подробное руководство под названием “Все, что вам нужно знать о FFUF – Everything you need to know about FFUF” на которое будет опираться далее данный обзор. В списке источников данное руководство указано под номером 2.

Ffuf это инструмент предназначенный для: веб-анализа с открытым исходным кодом, предназначенный для обнаружения элементов и контента в веб-приложениях или веб-серверах. [12]. То-есть, это инструмент, который поможет найти на сайте скрытые от пользователя файлы.

Основная функция, для которой ffuf используется это перебор каталогов сайта цели. Ещё хорошим примером будет часто, когда пользователь находит каталог, может быть необходимо найти его расширения. Это может быть полезно для поиска ошибок, когда есть zip-файл или файл резервной копии с таким же именем. [13]

Причины, по которым выбран этот инструмент следующие: скорость, гибкость и хорошая интегрируемость в другие инструменты, что является его одним из основных преимуществ, и также, потому что ffuf поддерживается сообществом и очень активно развивается его основателем.

Данный инструмент был добавлен в приложение и показал хорошие результаты по поиску скрытых файлов и директорий. Так на одной из тестовых web-приложений (demo.testfire.net – уязвимый сайт несуществующего банка) удалось найти базу данных, содержащую в себе логины и пароли пользователей, при отсутствии прямых ссылок на эту базу данных. Также была найдена admin панель защищённая паролем. Но учитывая найденные логины и пароли, удалось войти под правами администратора.

Данный OpenSource инструмент себя хорошо зарекомендовал среди пен тестеров, и имеет большое сообщество, которое его развивает.

Данный инструмент был успешно внедрён в итоговую версию приложения осуществляющего динамическое автоматизированное сканирование сетевого периметра цели.

Поиск скрытых файлов и директорий осуществляется на всех найденных поддоменах, самом целевом домене и всех найденных ip-адресах периметра сети целевого домена.

Процесс сканирования происходит параллельно. За счёт этого увеличивается скорость работы ffuf по поиску скрытых файлов и директорий в периметре сети цели по сравнению с последовательным перебором каждого найденного поддомена и ip-адреса.

### **2.1.3 Nmap**

Следующий инструментом, который будет здесь рассмотрен это Nmap предназначенный для сканирования сетей. Например, он может дать информацию о том, какие приложения запущены на сервере, какие порты в системе являются открытыми. Для рассмотрения данного инструмента будет использоваться официальный сайт разработчика, ссылка в списке источников под номером 19.

Nmap – довольно гибкий инструмент. Это включает в себя множество механизмов сканирования портов (как TCP, так и UDP), определение версии, проверку ring и многое другое [14]. Так вот nmap справится с этой задачей довольно быстро. Он был разработан для быстрого сканирования больших сетей, хотя прекрасно работает и с одиночными хостами [15].

Сам инструмент разрабатывался для быстрого сканирования действительно больших сетей. Это может быть полезно, так как даже в очень большой сети (около 16 миллионов IP адресов), активными могут быть лишь тысяча машин.

Nmap может предоставить информацию о версии программного обеспечения, когда было запрошено определение версии, предоставляет информацию о поддерживаемых IP-протоколах, предоставляет информацию

об обратных DNS-именах, предоставлять предположения об операционной системе, тепе устройства, а также MAC-адреса. По умолчанию Nmap выполняет только тщательную проверку, такую как сканирование портов, определение версии или определение операционной системы, на тех хостах, которые, как установлено, работают [16]

Таким образом, nmap может:

- Сканировать порты и затем показывать открытые порты,
- Предоставлять все сервисы, запущенные на найденных открытых портах,
- Определять версию запущенных приложений,
- Строить предположения об операционной системе
- Предоставлять информацию о поддерживаемых IP-протоколах, об обратных DNS-именах, MAC-адресах.
- Предоставлять информацию о типе устройства.
- Предоставлять информацию о типах брандмауэров. И этим список его возможностей можно сказать только начинается. Nmap в десятке (из 30,000) программ в репозитории Freshmeat.Net [17]

Nmap использовался в данном приложении для поиска открытых портов на найденных ip-адресах, а также осуществляет работу по поиску информации о версиях операционных систем (ОС) на которых функционирует целевой ресурс, а также nmap ищет информацию о версиях используемых сервисов и версиях программного обеспечения (ПО) работающих на целевом ресурсе.

Так, например, в результате сканирования цели nmap нашёл название и версию web-сервера используемого на целевом ресурсе.

#### **2.1.4 Nuclei**

Обновление Nuclei Данный инструмент можно назвать замыкающим. Так как он может использовать информацию собранную перечисленными ранее инструментами. Для выполнения собственно той задачи, которая стоит

передо мной, а именно обнаружение уязвимостей. На данном этапе имеется минимум следующее представление о сканируемой сети:

1. Список белых IP-адресов/сетей (subfinder)
2. Список DNS-имен, на которые отвечают веб-приложения (subfinder)
3. Список открытых портов, (nmap)
4. Список протоколов, используемых на открытых портах (nmap)
5. Версии используемых ОС, (nmap)
6. Версии используемого ПО (nmap)
7. Список скрытых файлов и каталогов (ffuf)

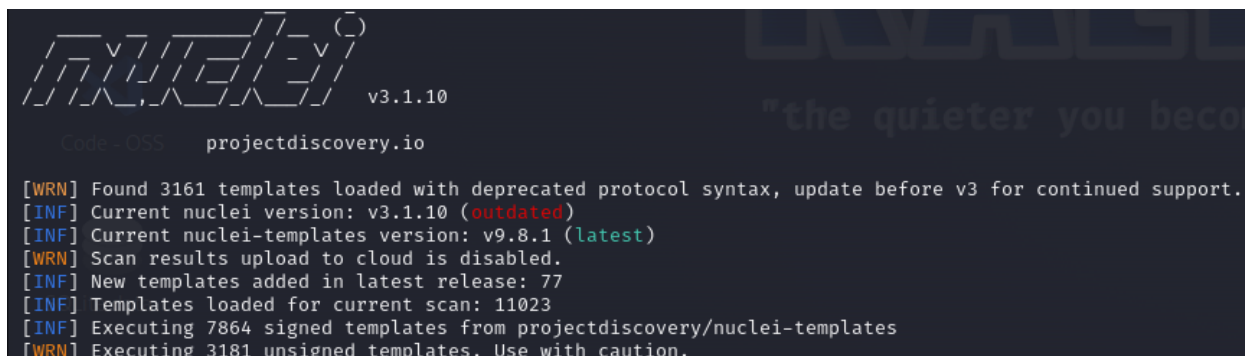
Описанию nuclei будет опираться на официальную страницу GitHub инструмента Nuclei. В списке источников она указана под номером 23. Также при написании данный обзор будет ссылаться на руководство, расположенное на официальном сайте разработчика данного инструмента. Называется это руководство “The Ultimate Guide to Finding Bugs With Nuclei” и ссылка на него в списке источников указана под номером 24.

Nuclei может смоделировать все виды проверок безопасности. Nuclei используется для отправки запросов между целевыми объектами на основе шаблона, что приводит к нулевому количеству ложных срабатываний и обеспечивает быстрое сканирование на большом количестве хостов. Nuclei предлагает сканирование по различным протоколам, включая TCP, DNS, HTTP, SSL, File, Whois, Websocket, Headless и т.д. [18].

Шаблон Nuclei представляет собой файл YAML. Данные разметки в файле указывают nuclei, что отправлять хосту и на что обращать внимание в ответе хоста, чтобы определить, уязвим ли он для определенной проблемы. [19]

Nuclei легко интегрируется в другие рабочие процессы инструмента. Является довольно быстрым и способен обрабатывать тысячи хостов за несколько минут. Также он отлично подходит для автоматизации, так с помощью него можно преобразовать ручные этапы оценки автоматизировав их путём написания шаблонов для каждой конкретной уязвимости и запускать

повторного тестирования с помощью этих шаблонов любое число раз на любое число хостов. На момент написания работы Nuclei содержит 11023 шаблонов. На 6 строчке рис. 1 видно точное число шаблонов на момент написания работы.



```

nuclei v3.1.10
Code - OSS projectdiscovery.io

[WRN] Found 3161 templates loaded with deprecated protocol syntax, update before v3 for continued support.
[INF] Current nuclei version: v3.1.10 (outdated)
[INF] Current nuclei-templates version: v9.8.1 (latest)
[WRN] Scan results upload to cloud is disabled.
[INF] New templates added in latest release: 77
[INF] Templates loaded for current scan: 11023
[INF] Executing 7864 signed templates from projectdiscovery/nuclei-templates
[WRN] Executing 3181 unsigned templates. Use with caution.
```

Рисунок 1 – Число шаблонов в Nuclei.

И даже не написав ни одного шаблона можно всего несколькими короткими командами в терминале, длинной в одну, две строки, проверить уязвимости, описанные в 11 тысяч шаблонах.

Nuclei, работает быстро, практически без ложных срабатываний изначительно упрощает работу за счёт автоматизации с помощью шаблонов. Все преимущества, описанные в данном обзоре, выделяют Nuclei среди конкурентов.

В процессе выполнения данной ВКР работы и по результатам работы полученного приложения nuclei, при поиске всех типов уязвимостей всех видов критичности просканировал цель за 45 минут.

## 2.2 Сохранение полученных результатов в таблицы базы данных

В приложение было добавлено сохранение полученных результатов в базу данных (БД) SQL. Для этого использовалась SQLite. Автоматизация процесса сканирования по прежнему осуществляется с помощи bash скриптов. Скрипты дорабатывались на основе документации bash [20].

За процесс сбора информации о ip-адресах целевой сети отвечает nslooper. При разработке была использовалась официальная документация [21].

За процесс поиска скрытых директорий и файлов отвечал фазер ffuf. Разработка опиралась на официальный проект ffuf [22]

За поиск открытых портов и определение используемых на стороне цели имён и версий ОС и ПО отвечал сканер nmap. Доработка скрипта с использованием nmap опиралась на официальную документацию [23].

За процесс поиска уязвимостей на найденных открытых портах отвечает nuclei. Доработка скрипта опиралась на официальную документацию [24]. А внедрение тегов и критичности уязвимостей опиралась на официальную GitHub страницу проекта со всеми шаблонами и подробным описанием этих шаблонов [25].

Процесс сохранения в БД собранной OpenSource инструментами информации происходит при помощи python скриптов. Процесс написания python скриптов опирался на документацию SQLite [26] официальную документацию python [27] и книги Марка Лутца “Изучаем Python” том 1 [28] и том 2 [29].

Полученная БД располагается по пути от корневого каталога: ./дата\_время\_имя-домена/inf/scan\_results.db

Всю собираемую информацию можно разделить на 6 групп и каждая из этих групп сохраняется в свою таблицу БД:

1. Таблица found\_files представляет список скрытых директорий и файлов, найденных для поддоменов целевого домена.
2. В таблице ip\_ports содержится информация о открытых портах для обнаруженных IP-адресов.
3. Таблица os\_info содержит сведения о версиях операционных систем, обнаруженных на сканируемом ресурсе. Эти данные могут быть полезны, если обнаружены версии операционных систем и программного обеспечения, для которых ранее были найдены уязвимости пентестерами.
4. В таблице services содержится информация о версиях программного обеспечения, включая данные о используемом веб-сервере.

5. Таблица `sub_domains` содержит обнаруженные поддомены целевого домена.

6. Таблица `vulnerability` содержит информацию о найденных уязвимостях, включая их критичность и подробное описание.

Таким образом в процессе сканирования цели и сбора информации создаётся 6 таблиц в которые записывается собранная информация.

Собранную информацию можно в дальнейшем использовать в своих целях для анализа или вывода собранной информации в своём проекте, который будет отображать информацию из БД, или использовать собранную информацию в этом же приложении для дальнейшего процесса автоматизации и так далее.

### **2.3 Замена Фреймворка flask на Django**

Было принято решение использовать Фреймворк `django`, а не `flask`. Это было сделано из-за популярности фреймворка и перспектив развития проекта, потому что изначально `flask` был выбран так как предполагалось, что полученное приложение будет довольно простым и перспектив развития у него не будет.

При переходе на Фреймворк `django` проект стал более легко поддерживаемый и масштабируемый. Приложение работает также корректно как и на `flask`.

В графический интерфейс была добавлена кнопка “RestartServer” перезапускающая сервер. На рис. 2 представлен графический интерфейс.



# Сканер уязвимостей

Введите доменное имя цели

Теги	Критичность
<input type="checkbox"/> CVE	<input checked="" type="checkbox"/> Info
<input type="checkbox"/> Panel	<input type="checkbox"/> High
<input type="checkbox"/> Wordpress	<input checked="" type="checkbox"/> Medium
<input type="checkbox"/> XSS	<input type="checkbox"/> Critical
<input checked="" type="checkbox"/> Exposure	<input checked="" type="checkbox"/> Low
<input checked="" type="checkbox"/> WP-Plugin	<input type="checkbox"/> Unknown
<input type="checkbox"/> OSINT	
<input checked="" type="checkbox"/> Tech	
<input type="checkbox"/> LFI	
<input checked="" type="checkbox"/> EDB	

Set Parameters      START      Restart Server

Рисунок 2 – Графический интерфейс

Сверху идёт поясняющее название. Далее поле для ввода доменного имени цели. И два столбца с флагами. Первый столбец обозначает тип искомой уязвимости. Второй столбец обозначает критичность искомой уязвимости. Имеется возможность выбрать все типы уязвимостей и все типы критичности уязвимостей и тогда сканер будет искать все возможные уязвимости цели с любой критичностью. Подробнее о типах критичности и тегах написано в работе прошлого семестра [30].

### **2.3.1 Изменение в работе сервера**

В процессе проведения экспериментов было принято решение заменить фреймворк flask на django. Это единственное, изменение произошло в структуре проекта. Так сам python сервер был перенесён в файлы views.py и urls.py с указанием структуры и путей GET и POST запросов для передачи введённых пользователем начальных параметров и начала работы процесса сканирования.

Html страница также была перенесена в стандартную директорию templates приложения django. А стили css в стандартную директорию static.

### **2.3.2 Изменение в работе скрипта**

В процессе написания работы были реализованы разные версии скрипта и в итоге было принято решение разбить скрипт на несколько частей. Каждая часть скрипта отвечала за работу отдельного OpenSource инструмента.

Коэффициент Скрипт состоит из нескольких bash скриптов и нескольких python скриптов.

#### **Шаги:**

##### **Запускается скрипт script.sh**

1. Запускается процесс сбора поддоменов с помощью subfinder.
2. Записываются собранные поддомены в Базу данных запуском python скрипта recording\_sub\_domains.py
3. С помощью nslookup запускается процесс поиска ip адресов для всех найденных поддоменов. Для этого используется цикл for перебирающий все найденные поддоменов.
4. Запускается в фоновом режиме скрипта dirrectory\_search.sh осуществляющего поиск скрытых деррикторий и файлов при помощи ffuf.
5. Запуск поиска открытых портом для каждого найденного ip-адреса при помощи nmap. Для этого также используется цикл for перебирающий все найденные поддомены.

6. Сохраняем полученные данные о ip и открытых портах на них в БД.  
Для этого запускается скрипт `recording_ip-ports.py`
7. Запускается процесс сканирования найденных открытых портов на наличие уязвимостей при помощи `nuclei`. Для этого запускается скрипт `vulnerability_search.sh`
8. Запускаем сканирование найденных открытых портов на наличие версий ОС и версий ПО. Для этого запускается скрипт `skanirovaniya_domenov_na_versii_OS-PO.sh`
9. В конце скрипта указываем параметр `wait` для ожидания завершения всех запущенных фоновых процессов. Иначе `docker` завершает свою работу не дождавшись завершения фоновых процессов сканирования, а завершает работу сразу после завершения работы скрипта `script.sh`

#### **Работа скрипта `dirrectory_search.sh`**

1. В цикле `while` для каждого ip-адреса и поддомена, а также целевого домена осуществляется запуск скрипта `ffuf_scan_domin-ip.sh` в фоновом режиме.
2. В конце скрипта указываем параметр `wait` для ожидания завершения всех запущенных фоновых процессов.

#### **Работа скрипта `ffuf_scan_domin-ip.sh`**

1. Запускается процесс сканирования для конкретного поддомена с помощью `ffuf`.
2. Запускается скрипт `recording_ffuf.py` в фоновом режиме для сохранения в БД полученных результатов.
3. В конце скрипта указывается параметр `wait` для ожидания завершения всех запущенных фоновых процессов.

#### **Работа скрипта `vulnerability_search.sh`**

1. Запускается процесс поиска уязвимостей на всех найденных открытых портах при помощи `nuclei`.

2. Запускается скрипт `recording_vulnerability.py` в фоновом режиме для сохранения в БД полученных результатов.
3. Работа скрипта `skanirovaniya_domenov_na_versii_OS-PO.sh`
  1. `scan_domain()` – функция для запуска сканирования домена и сохранения PID процесса в массив. Запускается в цикле `while` для каждого найденного открытого порта на каждом `ip` адресе. Для этого используется инструмент `nmap`.
  2. `check_nmap_process()` – функция для проверки завершения процесса сканирования. В случае завершения процесса сканирования вызывает скрипт `recording_OS-PO_BD.py` сохраняющий данные в БД.
  3. В цикле `for` происходит ожидание завершения всех процессов сканирования. Это гарантирует завершение всех процессов сканирования.
4. Имеются 5 python скриптов: `recording_ffuf.py`, `recording_ip-ports.py`, `recording_OS-PO_BD.py`, `recording_sub_domains.py`, `recording_vulnerability.py`. Все python скрипты выполняют аналогичную задачу по сохранению полученных результатов в БД. Во всех этих файлах есть 3 функции отвечающие за процесс создания базы данных, за процесс создания таблиц для хранения данных и заполнения этих таблиц данными.
  1. `create_database()` – создание базы данных SQLite и соответствующих таблиц для bash скриптов из которых этот python скрипт был вызван для создания этой таблицы. Эта функция вызывается первой.
  2. `parse_ffuf_output()` – парсинг данных. Изначально все OpenSource инструменты сохраняют свои данные в txt файлы. Это их стандартный способ сохранения собираемой информации, если не включить сохранение собираемой информации, то эта информация просто будет выводиться тем

же текстом в терминал. Таким образом данная функция парсит собранные данные из txt файла и сохраняет собранную информацию в python объектах (списках и словарях). Второй вызывается эта функция.

`insert_data()` – Отвечает за вставка данных в таблицы базы данных. Она пользуясь созданными функцией `parse_ffuf_output()` python объектами записывает полученные данные в базу данных. Эта функция вызывается последней.

## 2.4 Использование Docker

Для достижения кроссплатформенности и изолированности приложения, а также функционирования самого приложения на базе желаемого образа операционной системы, было принято решение использовать docker контейнер.

Чтобы запустить docker-compose из скланированного репозитория нужно выполнить следующие шаги:

1. Выполнить сборку образа выполнив в терминале из деректории проекта команду: `docker-compose build`
2. Узнать название собранного образа например `test_web_4-combined_container`. Выполните команду в терминале:

`docker images`

3. Запустить контейнер выполнив команду в терминале:

`docker run -v ./script -p 8000:8000 test_web_3-combined_container`

4. Web-приложение будет доступно по адресу:

`http://127.0.0.1:8000/semestr_3/`

Разработка опиралась на официальную документацию docker [31].

В качестве базового образа был выбран `kalilinux/kali-rolling`. Выбор был сделан в пользу данного образа из-за его легковесности (всего 53,32 МБ на момент написания работы), а также лёгкости установки всех необходимых

OpenSource инструментов из-за наличия всех необходимых инструментов в стандартном списке пакетов kali-linux.

Таким образом был разработан Docker образ, на базе образа kalilinux/kali-rolling, в который были прописаны необходимые параметры и благодаря этому при сборке образа происходит загрузка всех необходимых инструментов и зависимостей необходимых для функционирования приложения.

В docker-compose файле были прописаны параметры проброса порта 8000 таким образом, что порт 8000 внутри docker контейнера доступен из вне на ПК где был запущен данный контейнер. Подробнее про это написано на GitHub странице самого приложения [31].

Docker контейнер, созданный из получившегося docker образа, справился с поставленной задачей и изолировал работу приложения предоставив доступ к самому web-приложению вне контейнера.

## **2.5. Обзор аналогов Open Source инструментов, используемых в работе, но не вошедших в данную работу**

В данной глав 2.5 представлены результаты исследований компании positive technologies на которые опирались исследования данной работы. А исследования данной работы были необходимы для понимания направления разработки и соответственно выбора списка необходимы OpenSource инструментов.

По результатам исследования компании positive technologies в 96% организаций потенциальный злоумышленник может преодолеть сетевой периметр и проникнуть во внутреннюю сеть.

Оставшиеся 4% — это одна организация из банковского сектора, где в ходе работ удалось получить доступ в демилитаризованную зону (ДМЗ), то есть в буферную зону между внешней и внутренней сетью. Результаты представлены на рис. 3.

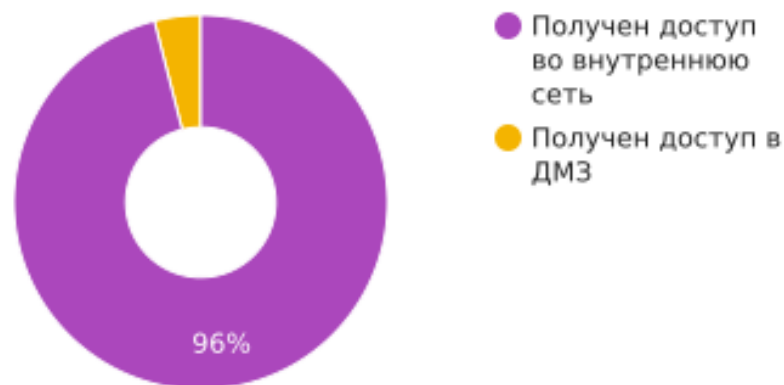


Рисунок 3 – Результаты исследований positive technologies

В 57% организаций существовал вектор проникновения, состоявший не более чем из двух шагов, но в среднем для этого потребовалось бы четыре шага. Результаты приведены на рис. 4.

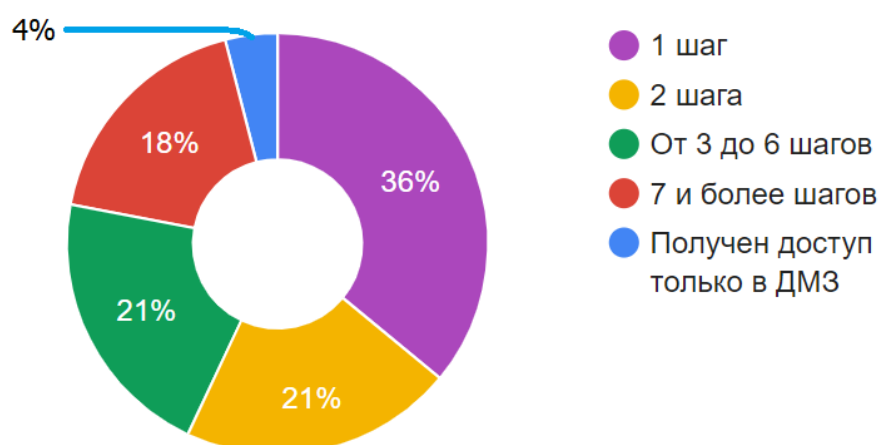


Рисунок 4 – Минимальное число шагов для проникновения в локальную сеть (доли организаций)

В данной работе будет осуществляться поиск всех представленные на рис. 5. уязвимостей. Статистика по результатам исследований компании positive technologies представлена на рис. 5.

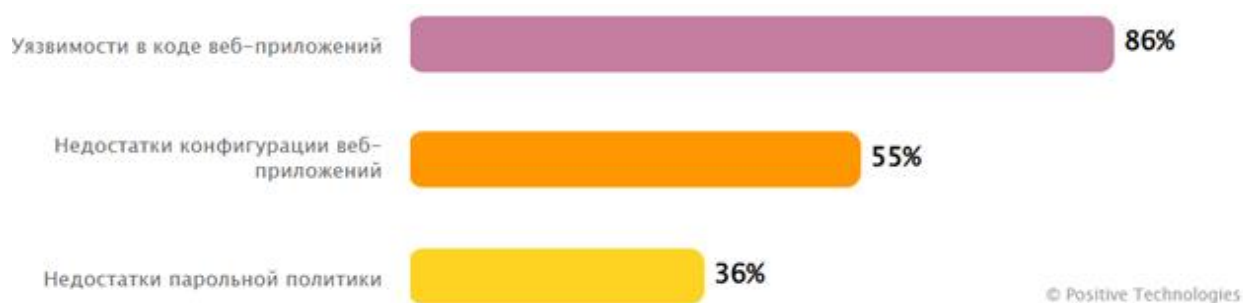


Рисунок 5 – Недостатки защиты, которые позволили получить доступ в ЛВС (доли организаций).

Большая часть типов уязвимостей и действий представленных на рис. 6 была реализована в рамках созданного приложения, полученного в процессе выполнения данной работы.

Поэтому имеет смысл опираться на подобные исследования других компаний.

По результатам исследования компании positive technologies были получены следующие данные. Среди легитимных действий чаще всего встречались действия, направленные на сбор данных об узлах сети и об уязвимых версиях ПО (доля таких действий среди всех атак составила 36%). На втором месте загрузка файлов (17%); под файлами подразумевается программное обеспечение, которое требовалось пентестерам для развития атаки. На третьем месте удаленное подключение к ресурсам организации (12%).





Рисунок 6 – Легитимные действия в системе в рамках внутренних пентестов (доли среди всех атак)

На основании этих данных, в процессе выполнения работы, было принято решение исследовать рынок инструментов, сравнить и использовать оптимальные для данной работы OpenSource инструменты.

Далее представлены OpenSource инструменты, от которых, в процессе выполнения работы было принято решение отказаться, а также объяснены сравнительные причины по которым было принято такое решение, не смотря на популярность, мощность и гибкость этих инструментов.

### **2.5.1 Amass**

Этот инструмент был разработан чтобы для стороннего наблюдателя или владельца показать как выглядит организация в интернете для поиска и перебора поддоменов DNS и составления карты сети на основе собранной информации. В процессе написания обзора данного инструмента будет опора на подробное руководство “How to Use OWASP Amass: An Extensive Tutorial” [32] автор Nick Gkogkos, а также руководством “An Extensive Tutorial” [33]. Отсылка к данным руководствам находятся на официальной странице GitHub проекта Amass и указана в списке источников под номером 3.

Amass можно использовать для следующей задачи. discover majority of an organisation’s externally exposed assets [32].

#### **Почему Amass ?**

Как написано в руководстве: Большинство доступных инструментов не являются полными, и полагаться только на них может создать ложное ощущение безопасности или привести к потере уязвимых ресурсов. В отличие от amass. [33]. Amass is backed by OWASP[33]. И поддомены организации (цели) могут находиться как в социальных сетях, HTTP-заголовках, в репозиториях исходного кода и так далее. Amass поддерживает пассивный и так и в активной конфигурации. Amass инструмент который обладает активной поддержкой, а это значит что он будет развиваться и если в нём имеются ошибки, то они будут устранены в будущем. Кроме того, скорость внедрения Amass высока, что потенциально означает лучшую согласованность данных и интеграцию с другими инструментами. [33].

Ещё Amass настраивается с помощью конфигурационного файла, что упрощает его обслуживание, упрощает интеграцию со сценариями.

## **Пассивный перебор**

Рассмотрим подробнее пассивный перебор. В данном случае перебор осуществляется по открытым источникам.

Поддерживает 55 источников: На момент написания этой статьи приложение поддерживало 55 источников, таких как API и веб-сайты, в рамках методов обнаружения поддоменов и сбора информации. [32]

## **Активный перебор**

Amass использует следующие методы перебора информации для перечисления DNS:

### **Возможности Amass**

Инструмент Amass обладает 5 главными функциями:

1. amass intel – Discover targets for enumerations [32]
2. amass enum – Perform enumerations and network mapping [32]
3. amass track – Track differences between enumerations [32]
4. amass viz – Visualize enumeration results [32]
5. amass db – Manipulate the Amass graph database [32]

Рассмотрим некоторые из них.

### **amass intel**

Amass выполняет самостоятельно огромную часть работы, но сам специалист всё равно должен выполнять дополнительные проверки результатов. Так Amass предоставляет следующие методы:

1. Используйте утилиты для определения доменов (например, dig, nslookup) [32]
2. Выполните поиск в WHOIS для подтверждения сведений об организации [32]
3. Выполните поиск результатов, таких как родительские домены, в поисковых системах [32]

Ещё amass intel может: искать названия организаций с массой, которые могли бы возвращать идентификатор [32].

### **amass enum**

Эта команда одна из самых полезных в Amass. Amass enum позволяет выполнять перечисление DNS и сопоставление цели, чтобы определить область атаки, которой подвергаются организации [32]. Результаты перечисления сохраняются в базе данных graph, которая будет расположена в выходной папке Amass по умолчанию или в указанном выходном каталоге с флагом “-dir”. Это также относится к другим подкомандам Amass . [32]

### **Запуск Amass в активной или пассивной конфигурации.**

Как уже упоминалось ранее Amass можно запускать как в пассивной так и в активной конфигурации. Пассивный режим: не будет проверяться информация DNS, например, путем разрешения поддоменов [32], в следствии этого пассивный режим и является намного быстрее активного.

Так например используя флаг "-passive": не даст возможности включить многие методы или конфигурации, такие как разрешение DNS и проверка подлинности [32].

Если использовать Amass в активной конфигурации используя все возможные методы перечисления DNS, то получится более точный результат, то-есть обнаружить большее число ресурсов.

### **amass track**

Тоже очень полезная команда, которая даёт возможность для любой конкретной цели сравнить результаты между перечислениями. К примеру если мы хотим сравнить результаты между двумя последними перечислениями для домена owasp.org. В этом поможет команда подставленная ниже на рис. 7.

Для команды `amass track`, пожалуй нужно представить результат работы представленный на рис. 7 и приведённый в подробным руководством “How to Use OWASP Amass: An Extensive Tutorial” [32] для лучшего понимания полезности данной команды и наглядности результатов её выполнения.

```
$ amass track -config /root/amass/config.ini -dir amass4owasp -d owasp.org -last 2
-----
Between 11/15 17:39:08 2019 UTC -> 12/08 22:16:49 2019 UTC
and     11/22 19:08:18 2019 UTC -> 12/08 22:16:44 2019 UTC
-----
Found: docs.owasp.org 172.217.18.179
Found: owaspforce.owasp.org 172.217.18.179
Found: sl.owasp.org 172.217.18.179
Found: owasp4.owasp.org 198.101.154.205
Found: gapps.owasp.org 172.217.18.179
Moved: mail.owasp.org
      from 172.217.22.51
      to   2a00:1450:401b:804::2013
Moved: lists.owasp.org
      from 2604:a880:2:d0::2080:a001
      to   2604:a880:2:d0::2080:a001,178.128.177.22
Removed: my.owasp.org 2620:46:2000:16::68
```

Рисунок 7 – вывод команды

Те строки в которых есть ключевое слово “Found” обозначают поддомены которые небыли найдены в предыдущем перечислении.

## Amass DB

Вы можете использовать эту подкоманду для взаимодействия с базой данных `Amass graph`, либо используемой по умолчанию, либо указать флаг “`-dir`”. [32]

Так например с помощью `Amass DB` можно перечислить все различные перечисления, выбранного домена, с дальнейшим сохранением их в базе данных графа “`amass4owasp`”.

## Amass Viz

Подкоманда `Amass Viz` визуализирует собранную информацию из базы данных “`Amass graph`”. Результаты также могут быть импортированы в `Maltego` для дальнейшего анализа с использованием `OSINT` (Open-Source Intelligence). [32].

## **Автоматизация упражнений OWASP по обнаружению накоплений**

Довольно важным аспектом в работе является автоматизация исследования периметра для обнаружения уязвимостей.

Это актуально потому что поддомены организации могут быть активны или неактивны в разные промежутки времени.

Так в Amass перечисленные методы может использоваться в сочетании с периодическими мероприятиями по сбору информации и подсчету поддоменов [32]

Далее можно написать сценарий для отправки оповещений при обнаружении нового ресурса.

Вместе с Amass мы можем написать скрипт, например на GNU Bash, который при помощи мобильного приложения отправляет push-уведомления на телефон с помощью pushover.net API. Это достигается с помощью инструмента wget для выполнения запросов к API, которые отправляют уведомление на мой телефон о новых поддоменах [32]. Можно реализовать аналогичную функциональность, используя веб-приложения Slack или Discord. [32].

### **Краткий вывод об Amass**

В заключение рассмотрения данного инструмента можно сделать вывод, что для последующей работы в рамках ВКР данный инструмент слишком перегружен, из-за его большой функциональности. А так же его скорость работы значительно ниже используемого в работе аналога subfinder, который справился с задачей сбора под доменов в 34 раза быстрее amass.

### **2.5.2 Assetfinder**

На официальной странице GitHub почти совсем нет информации о возможностях данного инструмента. Ссылка в списке источников под номером 34. Это стало причиной по которой была собрана информация о возможностях данного инструмента из сторонних источников.

Этот инструмент, аналог Amass и Subfinder. Он разработан для получения потенциальных поддоменов целевого домена [34]. Как и subfinder этот OpenSource инструмент осуществляет поиск ресурсов в пассивном режиме не выдавая свою прямую активность на целевому хосту - это самый быстрый инструмент для возврата поддоменов целевого домена. [34]. Также по мнению следующего автора: Asset finder - это один из самых быстрых инструментов для поиска поддоменов, доступных на рынке. [35]

Assetfinder использует следующие ресурсы для поиска:

1. crt.sh [35]
2. certspotter [35]
3. hackertarget [35]
4. threatcrowd [35]
5. wayback machine [35]
6. dns.bufferover.run [35]
7. facebook [35]
8. virustotal [35]
9. Findsubdomains [35]

Он обладает всего одной подкомандой. Проще инструмента сложно представить. Однако он себя хорошо зарекомендовал среди специалистов по информационной безопасности. По мнению автора источника под номером 7 в списке источников, этот инструмент занял 2 место после Amass в top-5 инструментов перечисления поддоменов.

Этот инструмент, быстрее, он проще двух предыдущих в освоении и использовании.

Но в процессе выполнения работы было принято решение использовать subfinder из-за простоты его встраивания в процесс автоматизации при помощи bash скриптов.

### 2.5.3 Dirsearch

Ещё один фаззер как и ffuf. Данный обзор будет опираться на официальную страницу GitHub. Ссылка на неё в списке источников находится под номером 36. Используется для перебора скрытых веб-каталогов и файлов. Его можно назвать кроссплатформенным, так как работать он будет на таких операционных системах как GNU/Linux, Windows XP/7/8/10, MacOSX [36]. Используется этот инструмент для brute force директорий файлов на веб-сайтах. Его основными функциями можно выделить:

- Многопоточность,
- Рандомизация пользовательских агентов,
- Поддержка проху,
- Задержка запросов.

По сравнению с ffuf он довольно простой, но также является очень востребованным и сильным инструментом предназначенном для сканирования каталогов.

На официальном источнике оказалось очень мало информации. Предположительно это связано с высокой простотой данного инструмента и как следствие необходимо включить в данный обзор сторонние источники. Поддерживается возможность установки как минимального так и максимального размера файлов одновременно. Что даёт возможность указания необходимого в рамках конкретного поиска допустимого размера файлов.

Данный инструмент прост в использовании но при этом не является достаточно мощным поэтому выбор был сделан в пользу его аналога ffuf.



### **3 РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ**

#### **3.1 Полученные результаты по скорости работы приложения**

Сканирование целевого домена vulnweb.com со всеми внесёнными в скрипт изменениями заняло 15:42 секунды. За это время было найдено 23 поддомена целевого домена, было найдено 8 уязвимостей, 7 из которых критичности info, и 1 medium. Была собрана информация о ip-адресах и открытых портах на них. На открытых портах были найдены версии операционных систем и определены имя и версия используемого web-сервера на котором функционирует целевой домен. А также были найдена admin панель и sql файлы определяющие структуру базы данных используемых на сайте.

Сложность оценки времени сканирования заключается в самом процессе сканирования, который зависит от мощности ПК на котором запускается приложение, также от пропускной способности сети и от максимальной нагрузки которую может выдержать цель, так как все OpenSource инструменты используемые в данной работе настроены на автоматическую подстройку под цель, и для разных целей они будут работать с разной скоростью. Это сделано специально, чтобы не перегружать цель чрезмерным количеством параллельных запросов.

#### **3.2 Анализ полученных результатов**

Subfinder нашёл все поддомены цели за 4.5 секунды.

Nsloosur нашёл ip-адреса на которых располагаются все поддомены за 16 секунд.

Nmap нашёл открытые порты на найденных ip за 1 минуту 9 секунд.

В процесс динамического автоматизированного сканирования было добавлено использование такого инструмента как ffuf осуществляющего фаззинг, то-есть перебор и подстановку значительного числа запросов 20476 для каждого из найденных ip-адресов и поддоменов. За время работы инструмента ffuf было отправлено 552852 запроса за 1 час 32 минут.

Была максимально распараллелина работа используемых OpenSource инструментов. Все инструменты работают максимально параллельно, как только одним из OpenSource инструментом были получены необходимые данные для работы второго OpenSource инструмента, то сразу происходит процесс передачи полученных данных от одного инструмента в другие инструменты и запуск новых инструментов с использованием собранных данных. Однако как уже упоминалось выше по тексту подобное решение помогло найти значительные уязвимости на целевом домене, такие как база данных с логинами и паролями пользователей, admin панель и sql файлы определяющие разрешённые параметры пароля, что в дальнейшем можно использовать при подборе пароля тем же фазером ffuf.

Nuclei работал параллельно с ffuf, который отправлял большое число запросов и поэтому его работа была замедлена. Он отработал и нашёл 8 уязвимостей, 7 из которых критичности info, и 1 medium за 44 минуты, за это время было применено 11023 шаблонов nuclei.

Таким образом в процессе работы скрипта было найдено 23 поддомена, 8 уязвимостей, 3 ip адреса и 4 открытых порта на них. На открытых портах были найдены версии ПО функционирующих на найденных ip-адресах и имя и версию используемого web-сервера. Было найдено имя и версия ОС на которых работает сканируемое web-приложение и найденный web-сервер. Была найдена admin панель и SQL файлы определяющие параметры и структуры таблиц целевого ресурса.

Время работы всего скрипта отвечающего за динамическое автоматизированное сканирование цели составило примерно 1 час 33 минуты. За это время было отправлено порядка 600000 запросов на целевой ресурс.

Было проведено 5 запусков на целевом домене vulnweb.com и статистика запусков приведена на рис. 8, рис. 9, рис. 10, рис. 11, рис. 12, рис. 13. Для всех графиков по левой оси указаны секунды, а снизу номер запуска.

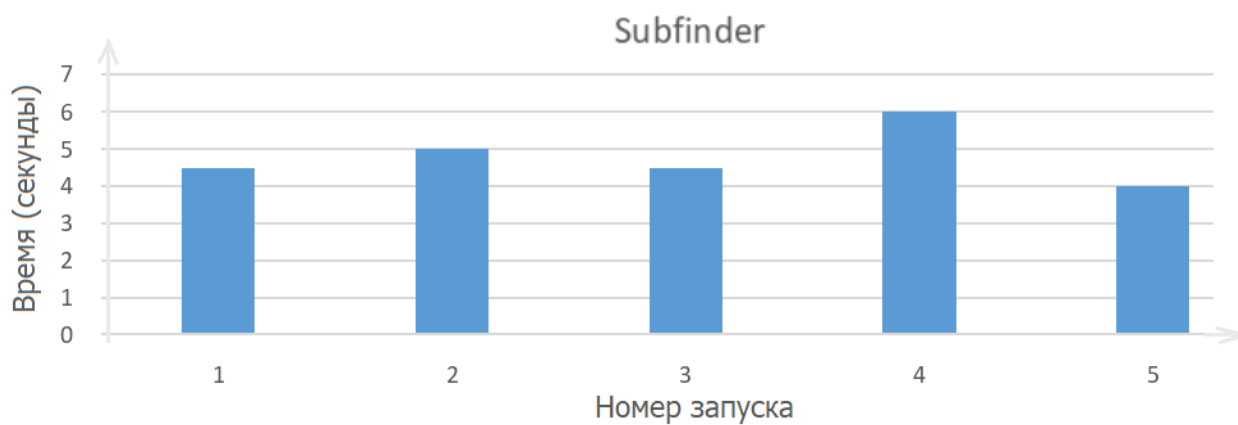


Рисунок 8 – Статистика запуска для OpenSource инструмента subfinder

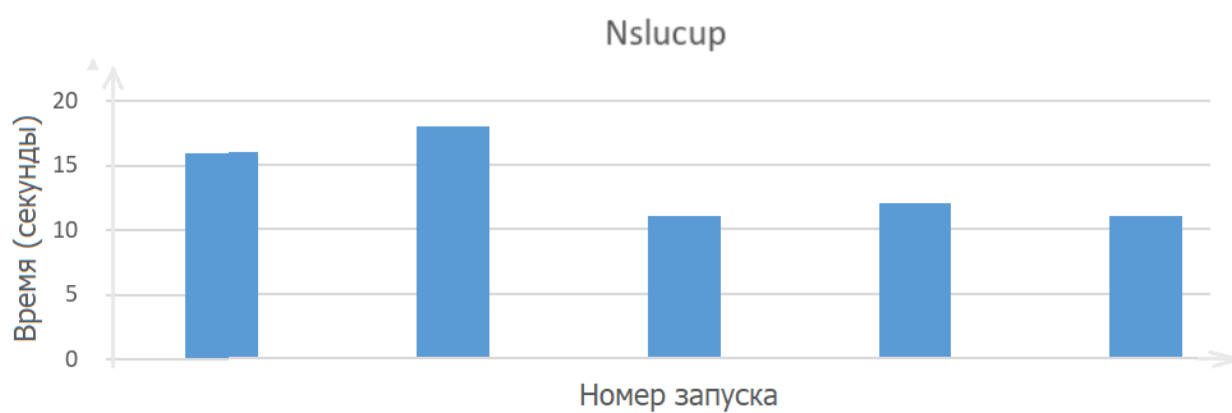


Рисунок 9 – Статистика запуска для OpenSource инструмента nsloocup

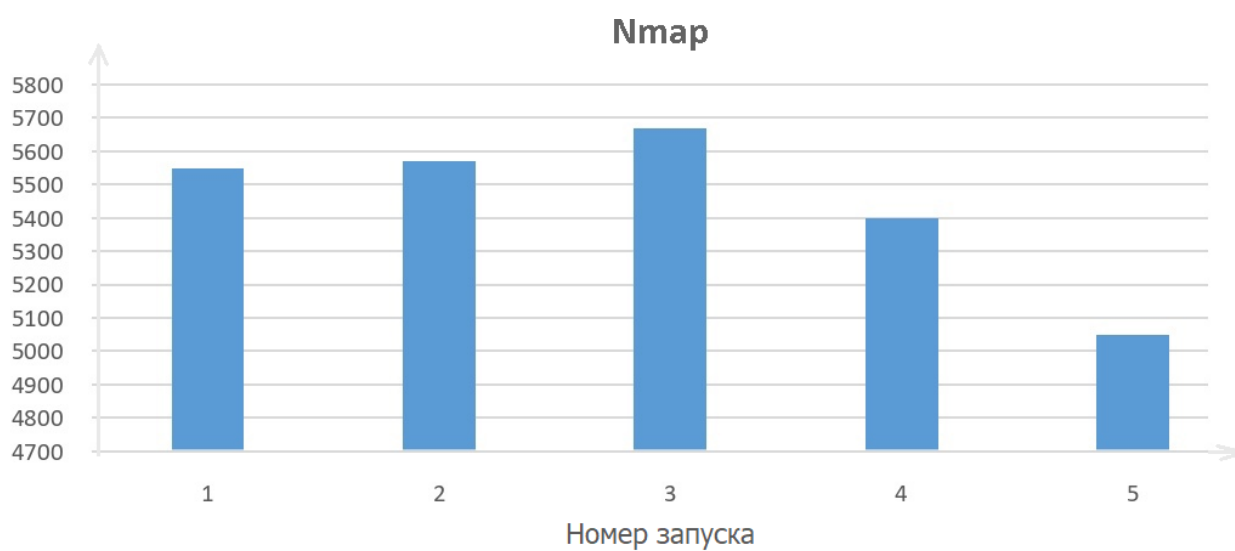


Рисунок 10 – Статистика запуска для OpenSource инструмента nmap

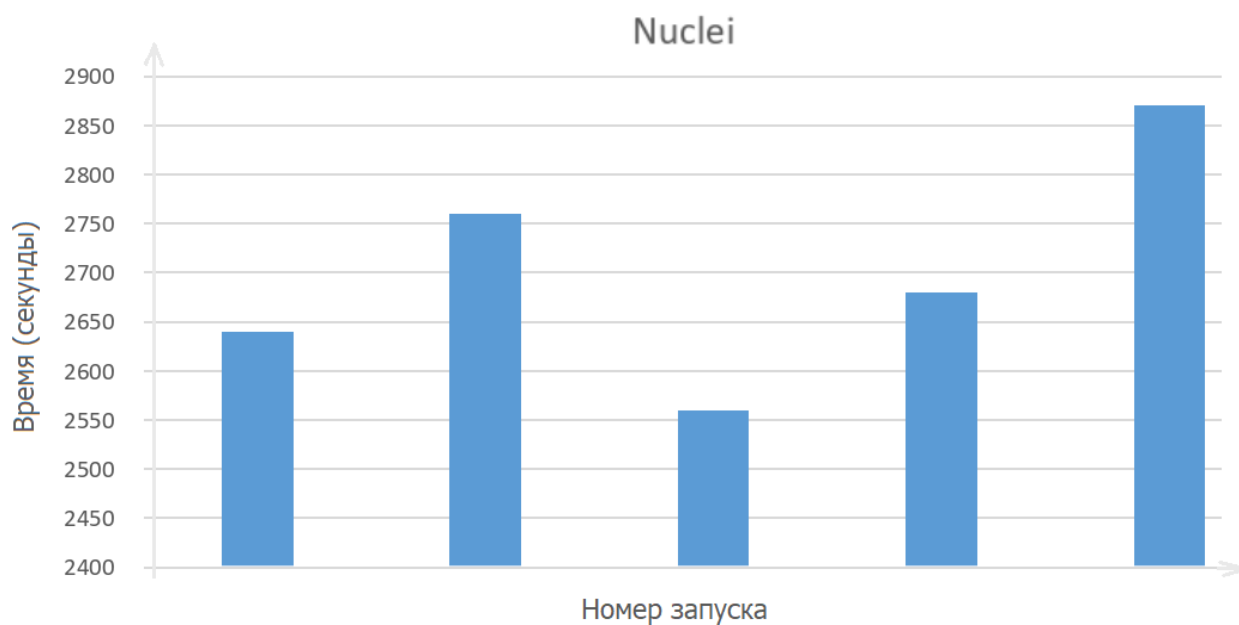


Рисунок 11 – Статистика запуска для OpenSource инструмента nuclei

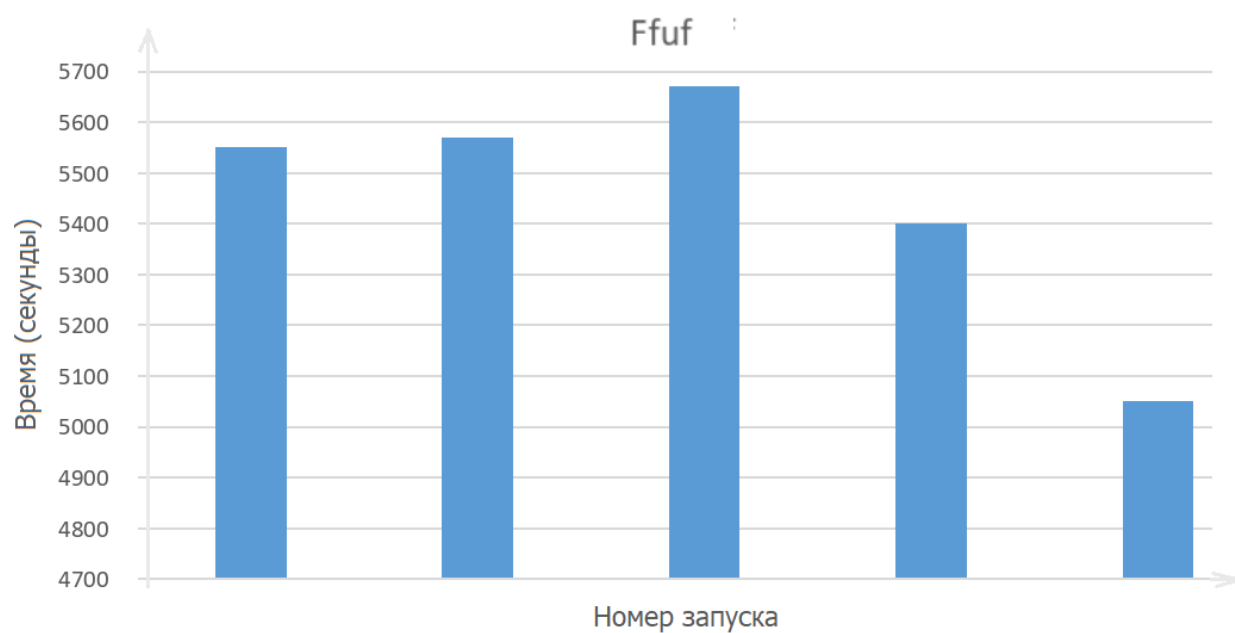


Рисунок 12 – Статистика запуска для OpenSource инструмента ffuf

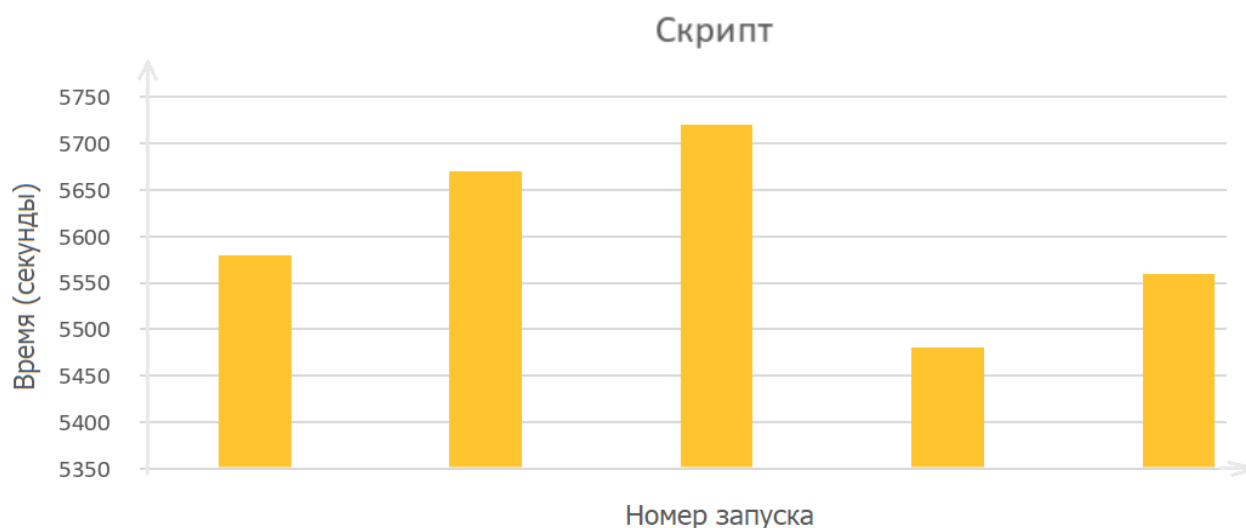


Рисунок 13 – Статистика запуска всего скрипта

Произведём запуск скрипта с указанием только трёх типов уязвимостей и только критических уязвимостей. Результаты представлены на рис. 14.



Рисунок 14 – Статистика запуска всего скрипта

Произвели запуск для второго ресурса целевого ресурса demo.testfire.net. Результаты запуска представлены на рис. 15.



Рисунок 15 – Статистика запуска всего скрипта

Произведём запуск скрипта с указанием только трёх типов уязвимостей и только критических уязвимостей. Результаты запуска представлены на рис. 16.

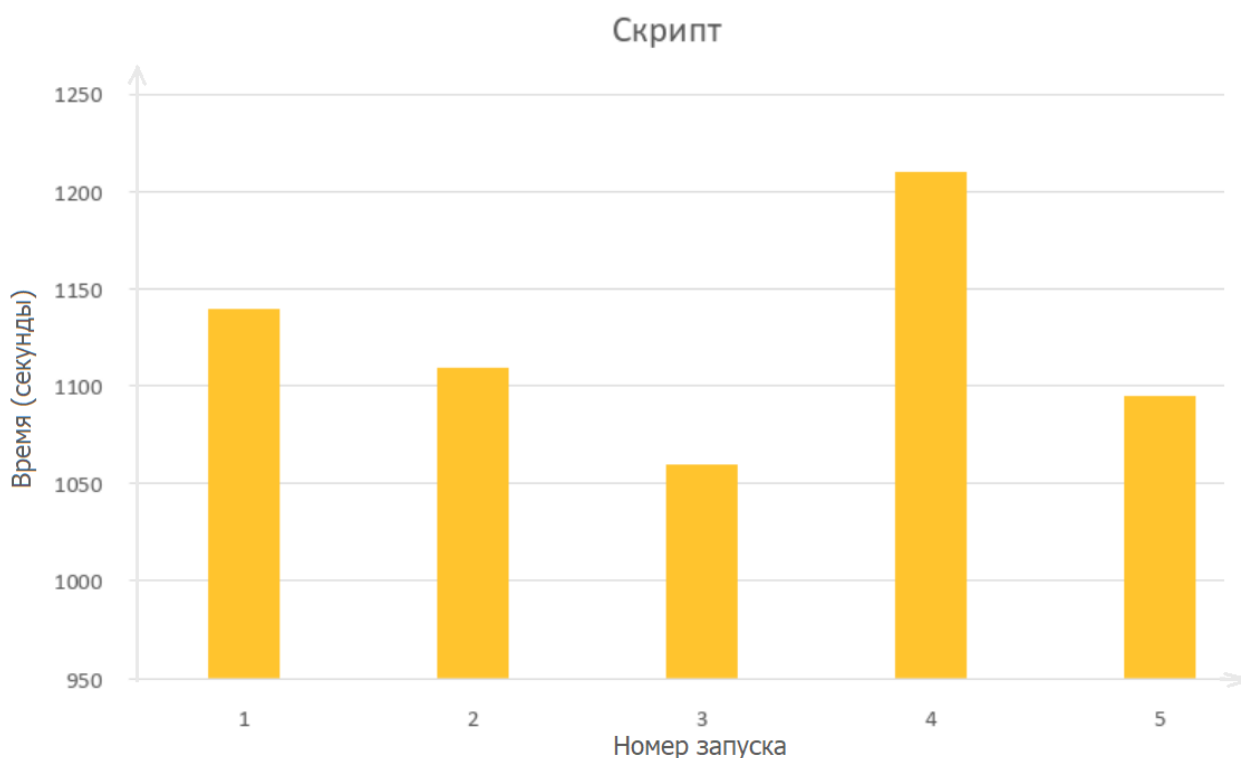


Рисунок 16 – Статистика запуска всего скрипта

### 3.3 Краткие выводы по полученным результатам

По полученным результатам можно сделать вывод о том, что за пару десятков минут данное приложение может составить картину представляющую периметр сети из доменных имён, и ip-адресов на которых эти доменные имена расположены. Найти список уязвимостей в периметре сети: открытые порты на найденных ip-адресах, скрытые файлы и каталоги сайта, найти имена и версии ПО и ОС на которых функционирует сканируемое web-приложение расположенное по целевому домену, а также найти список уязвимостей на найденных открытых портах.

### **3.4 Расположение результатов работы**

Само приложение доступно на удалённом репозитории по ссылке:  
<https://github.com/JackoPrograms/skaner.git>

В качестве удалённого репозитория был взят GitHub.

Данное действие добавило возможность получить доступ к приложению через простое клонирование репозитория на локальную машину.

Подробная инструкция по скачиванию приложения, из удалённого репозитория, сборке Docker образа, запуску и доступности приложения доступна по этой же ссылке в самом репозитории.

## **4 ОЦЕНКА И ЗАЩИТА РЕЗУЛЬТАТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ**

### **4.1 Описание результата интеллектуальной деятельности**

Право на интеллектуальную собственность присваивается автору в соответствии с Конституцией Российской Федерации (РФ), Гражданским кодексом РФ, федеральными законами, подзаконными нормативно-правовыми актами и международными договорами.

Под интеллектуальной собственностью понимается – некие результаты полученные в процессе интеллектуальной деятельности, а также средства индивидуализации юридических лиц (ЮЛ) и предпринимателей.

Существуют виды охраняемых результатов интеллектуальной деятельности и средств индивидуализации. В рамках данной работы к ним относятся:

1. Базы данных;
2. Программы для электронных вычислительных машин (программы для ЭВМ);

На основании статьи 1225 ГК РФ:

Как продукт, предназначенный для использования на ЭВМ, база данных (БД) имеет много общих черт с программным обеспечением для ЭВМ. Как и программы, БД является объектом интеллектуальной деятельности, подлежащим правовой защите в соответствии с законодательством (ссылка на соответствующую статью Гражданского кодекса Российской Федерации). Более того, разработка БД обычно осуществляется в рамках систем управления базами данных (СУБД), и операции с ней выполняются в этой же среде на протяжении всего ее существования. Следовательно, БД тесно связана с определенной компьютерной программой, и поэтому отдельно они имеют незначительную ценность.

В данной работе были написаны программы для ЭВМ состоящие из bash скриптов для операционной системы kalilinux. А также скрипты написанные на языке программирования python. Вместе осуществляющие сканирование



сетевого периметра цели на наличие уязвимостей, а также определяющие версии программного обеспечения (ПО) и версии операционных систем (ОС) на которых работает данный сетевой периметр. Полученные результаты при помощи python скриптов формируют БД и автоматически сохраняют полученные результаты в БД.

На основании статьи **1261** ГК РФ:

Можно сделать вывод, что правовая защита авторских прав на программное обеспечение для компьютеров, включая операционные системы и программные комплексы, аналогична защите прав на литературные произведения. Программа для компьютера представляет собой определенный набор данных и инструкций, предназначенных для использования на различных устройствах с целью достижения определенного результата. Эти права охватывают исходный код, объектный код, а также дополнительные материалы, включая аудиовизуальные элементы, созданные в процессе ее разработки.

Таким образом, программирование, как и любая другая форма творчества, защищено авторским правом в соответствии с Гражданским кодексом Российской Федерации. Элементы программирования бывают литературные и нелитературные:

- Литературные: написанный код программы.
- Нелитературные это интерфейс приложения, аудиоматериалы и видеоматериалы, дизайн, шрифты, логотипы, изображения, анимации и другие графические элементы, которые используются в программном продукте.

Защита программ также способствует развитию рынка программного обеспечения. Когда авторы уверены в том, что их интеллектуальная собственность будет защищена, они более склонны инвестировать в разработку новых продуктов. Это приводит к увеличению конкуренции и повышению качества программного обеспечения на рынке.

Литературное произведение – это результат творческого процесса, в котором автор вкладывает свой ум и время. Это акт творчества, требующий значительного умственного и временного вложения. Следовательно, защита авторских прав на литературные произведения является необходимой, чтобы признать и защитить труд и интеллектуальные усилия их создателей. Это относится не только к классическим литературным текстам, но и к программам для компьютеров, которые также могут быть признаны литературными произведениями.

Защита программного кода как литературного произведения имеет ключевое значение для стимулирования инноваций в сфере информационных технологий. Когда авторы программ знают, что их творческие усилия будут защищены законом, это способствует развитию новых технологий и продуктов. Это создает благоприятную среду для инноваций, поскольку авторы могут чувствовать себя уверенно в вопросах правовой защиты своих разработок.

Кроме того, защита программ как литературного произведения способствует уважению к интеллектуальной собственности в целом. Это помогает формировать культуру уважения к правам авторов и осознание важности интеллектуальной собственности для общественного развития. Создание и поддержание такой культуры способствует укреплению правовых норм и стимулирует творческую активность в различных областях, включая информационные технологии.

У разработчика программного продукта есть два вида прав: имущественные и неимущественные. Неимущественные права, такие как право на авторство и защиту репутации, не подлежат передаче третьим лицам. В отличие от этого, в имущественном аспекте автор имеет возможность осуществлять такие действия, как воспроизведение, продажа, передача по лицензии, внесение изменений и регистрация программного обеспечения или баз данных.

Таким образом в данной работе объектом исследования является полученная в ходе разработки программа для ЭВМ “Приложение обеспечивающее возможности объединения OpenSource инструментов в единый workflow и возможность их добавления”. И созданные полученным workflow БД для хранения полученных результатов.

Программа для ЭВМ представляет собой приложение, созданное на языках программирования Python, Bash, JavaScript, HTML и CSS, а также с использованием технологии контейнеризации Docker и фреймворка Django для веб-приложений на языке Python.

Данная разработка “Приложение обеспечивающее возможности объединения OpenSource инструментов в единый workflow и возможность их добавления” доступна для загрузки из удалённого репозитория через сеть интернет или на физических информационных носителях, таких как:

- Жесткие диски (Hard Disk Drives, HDD)
- Флеш-накопители (USB-флешки)
- CD и DVD диски
- Blu-ray диски
- Внешние жесткие диски
- Магнитооптические диски (МО-диски)
- Магнитные ленты (магнитофонные катушки, магнитные карты)
- Флоппи-диски (дискеты)
- Благодаря возможности клонирования приложения из удалённого репозитория обеспечивается целостность приложения при передачи и облегчить процесс установки.

Полученное приложение может использоваться как частными лицами, так и компаниями, для сканирования своего сетевого периметра или периметра цели при наличии соответствующих разрешений на сканирование.

На основании ранее предоставленной информации о принадлежности программ для ЭВМ и баз данных к интеллектуальной собственности, а также

принадлежности полученной разработки к средствам программ для ЭВМ и БД. Можно сделать вывод, что объектом данного исследования являются результаты выпускной квалификационной работы (ВКР), представляющие собой интеллектуальную собственность.

#### **4.2 Оценка рыночной стоимости результата интеллектуальной деятельности**

Выделяются три подхода при оценке рыночной стоимости результатов интеллектуальной деятельности: доходный, сравнительного (рыночного) и затратного.

Эти три подхода обладают следующими характеристиками:

Доходный подход состоит оценке объектов интеллектуальной собственности основывается на тщательном изучении прогнозируемых финансовых преимуществ, которые могут быть получены от использования оцениваемой интеллектуальной собственности. Этот подход использует концепцию ожидаемых результатов и основан на принципе учета будущих доходов и их капитализации. Суть метода заключается в определении текущей стоимости ожидаемых будущих доходов, которые могут быть получены благодаря эксплуатации объекта интеллектуальной собственности.

Затратный подход это подход к определению стоимости объекта интеллектуальной собственности путем анализа затрат, необходимых для его воссоздания или замены с учетом его текущего состояния, износа и устаревания. Этот метод основывается на предположении, что стоимость объекта можно оценить через расчет затрат на его воспроизводство или замену на сегодняшний день. При использовании метода затратной оценки учитывается не только стоимость материалов и труда, но и другие затраты, такие как интеллектуальные ресурсы, затраты на исследования и разработки, юридическую поддержку и т. д. Кроме того, учитывается износ и устаревание технологии или знаний, что также влияет на общую стоимость объекта. Этот метод чаще всего используется для оценки объектов интеллектуальной

собственности, которые находятся на стадии разработки или для которых сложно предсказать будущий доход. Например, для стартапов или новых технологий, у которых еще нет устойчивого потока доходов, метод затратной оценки может быть более подходящим, поскольку он оценивает фактические затраты на создание их стоимости, не зависящие от потенциального дохода.

Сравнительный (рыночный) подход к оценке стоимости объекта интеллектуальной собственности базируется на анализе цен аналогичных объектов, проданных на рынке. Оценщик при использовании этого метода ищет сравнимые объекты, которые уже были проданы или оценены, и использует их цены как основу для определения стоимости анализируемого объекта. Этот подход особенно ценен, когда имеется достаточное количество сравнимых сделок для проведения анализа, что позволяет получить более точную оценку стоимости.

Данный метод также предполагает корректировку цен сравнимых объектов с учетом различий между ними и оцениваемым объектом. Это позволяет учесть особенности и специфику каждого объекта, а также его уникальные характеристики, которые могут влиять на его стоимость. Таким образом, сравнительный подход обеспечивает более объективную оценку стоимости объекта интеллектуальной собственности путем сопоставления его с аналогичными объектами на рынке.

Организации или частные лица заинтересованные в приобретении результатов данной интеллектуальной деятельности заинтересованы в оценке рыночной стоимости этих результатов интеллектуальной деятельности, потому что планируют приобретать и использовать их. Разработчики также заинтересованы в проведении оценки рыночной стоимости.

Рыночная стоимость отражает наиболее вероятную цену, за которую объект интеллектуальной собственности может быть продан на открытом рынке в условиях соперничества.

В рамках данной работы будет рассмотрен затратный подход. Он был выбран исходя из текущего состояния объекта интеллектуальной собственности.

Применение рыночного подхода затруднено поиском аналогов выполняющих поиск именно таких уязвимостей. Это в свою очередь затрудняет сравнительную оценку полученного продукта с аналогами.

На данный момент продукт интеллектуальной собственности ещё не был введён в коммерческую эксплуатацию. Это затрудняет использование доходного подхода в данной работе.

Таким образом именно затратный подход является самым приемлемым для оценки стоимости данного проекта.

В соответствии с положениями Федеральных стандартов бухгалтерского учета (ФСБУ) 14/2022, оценка объектов нематериальных активов (НМА) осуществляется по их первоначальной стоимости. Под первоначальной стоимостью НМА подразумевается совокупность капитальных вложений, связанных с данным объектом, понесенных до момента признания его в бухгалтерском учете как нематериального актива. В состав первоначальной стоимости включаются фактические расходы на приобретение и создание НМА в соответствии с предписаниями ФСБУ 26/2020 (пункт 9). Данный подход обеспечивает достоверность и точность оценки стоимости НМА, учитывая все реальные затраты, связанные с его созданием и приобретением, и исключая случайные или косвенные расходы, не относящиеся к формированию актива.

Таблица №1 содержит список затрат, которые необходимы для создания программного продукта “Приложение обеспечивающее возможности объединения OpenSource инструментов в единый workflow и возможность их добавления”.

Таблица №1 — Затраты на ЗП работников за месяц

Должность	Оклад руб./месяц.	Рабочая норма Часов / месяц.	Часовая ставка работника руб / час.
Backend разработчик	150000, 00	160	937,5
Frontend разработчик	120000,00	160	750
Специалист по информационной безопасности (пентестер)	150000,00	160	937,5

На создание итогового приложения было затрачено 840 часов пентестером, Backend разработчик затратил 340 часов, а Frontend разработчик потратил 48 часов.

Получившиеся затраты приведены в таблице №2.

Таблица №2 — Затраты на ЗП разработчикам за всё время

Должность	Затрачено часов	Часовая ставка работника руб / час.	Сумма затрат на оплату труда
Backend разработчик	840	937,5	787080,00
Frontend разработчик	340	750	255000,00
Специалист по информационной безопасности (пентестер)	48	937,5	44976,00

Общие затраты на оплату труда составили  $787080,00 + 255000,00 + 44976,00 = 1\,087\,056,00$  руб.

В таблице под номером 3 приведены затраты на оборудование необходимое в процессе разработки.

Таблица №3 — Затраты на оборудование

Товар	Первоначальная стоимость оборудования, руб.	Срок полезного использования, мес.	Амортизация оборудования за месяц
Рабочий ноутбук backend разработчика: Ноутбук IdeaPad L340	87498,00	80	1093,76
Рабочий ноутбук Frontend разработчика: Ноутбук Xiaomi Book Pro 16	107375,00	80	1342,19
Рабочий ноутбук пентестера: Ноутбук Lenovo IdeaPad Slim	53735,00	80	671,69

Итоговая сумма затрат на оборудование составила  $87498,00 + 107375,00 + 53735,00 = 248608,00$  руб.

А сумма амортизации равняется  $1093,76 + 1342,19 + 671,69 = 3107,64$  руб.



Расчёты на программное обеспечение представлены в таблице под номером 4.

Таблица №4 — Затраты на программное обеспечение

Товар	Стоимость ПО, руб.	Срок полезного использования, мес.	Стоимость часа / руб.
PyCharm	2 268,97	1	14,18
Программное обеспечение: Windows 11	3500,00	36	0,6
WebStorm	1 448,86	1	9,1

PyCharm использовался 4,25 месяца одним разработчиком.

WebStorm 2.125 месяца одним разработчиком.

Windows 11 приобретался единожды для трёх разработчиков.

Итоговая сумма затрат на программное обеспечение равняется  $2268,97 * 4,25 + 3500,00 * 3 + 1448,86 * 2,125 = 23221,95$  руб.

Затраты на аренду помещения представлены в таблице под номером 5.

Таблица №5 — Затраты на аренду помещения

Предмет	Арендная плата помещения руб. / месяц	Арендная плата в час / руб.
Стоимость помещения (45 кв. м)	60000,00	375,00

Итоговая сумма затрат на аренду помещения равняется  $60000,00 * 4,25 = 255000,00$  руб.

Затраты на продукты питания представлены в таблице под номером 6.

Таблица №6 — Затраты на продукты питания

Товар	Первоначальная стоимость, руб.	Количество	Итоговая стоимость
Канистра для куллера	650,00	16	10400,00
Трёхразовое питание для сотрудников	2550,00	60	155340,00

Итоговая сумма затрат на продукты питания равняется  $10400,00 + 155340,00 = 165740,00$  руб.

Затраты на прочую технику и мебель представлены в таблице 7.

Таблица №7 — Затраты на прочую технику и мебель

Товар	Первоначальная стоимость оборудования, руб.	Количество	Итоговая стоимость
Куллер с водой	3150,00	1	3150,00
Рабочий стол	15000,00	4	60000,00
Рабочее кресло	12000,00	4	48000,00
Холодильник	30000,00	1	30000,00

Итоговая сумма затрат на прочую технику и мебель равняется  $3150,00 + 60000,00 + 48000,00 + 30000,00 = 141150,00$  руб.

Затраты на оплату счетов представлены в таблице под номером 8.

Таблица №8 — Затраты на оплату счетов

Предмет	Стоимость тарифа в руб./ месяц.	Стоимость 1 часа услуги – интернет, руб.
Услуги по предоставлению доступа в сеть интернет	850,00	5,31
Оплата электроэнергии	2000,00	12,5

Итоговая сумма затрат за 4,25 месяца равняется  $850,00 * 4,25 + 2000 * 4,25 = 12112,5$  руб.

Расчёт общих затрат на разработку приложения приведены в таблице под номером 9.

Таблица №9 — Общие расходы:

Категория затрат	Затраты, руб.
Оплата труда	1 087 056,00
Амортизация оборудования	3107,64
Программное обеспечение	23221,95
Аренда помещения	255000,00
Продукты питания	165740,00
Прочая техника и мебель	141150,00
Оплата счетов	12112,5
Итого	1 687 388,09

Итого, можно сделать вывод, что общие затраты на создание программного продукта составляет 1 687 388,09 рублей

### **4.3 Правовая защита результатов интеллектуальной деятельности**

Для обеспечения юридической защиты разработки необходимо пройти официальную процедуру ее регистрации в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам (Роспатент). Этот процесс гарантирует признание прав на интеллектуальную собственность и обеспечивает защиту от несанкционированного использования или копирования со стороны третьих лиц. Регистрация программы в Роспатенте подтверждает ее авторство и устанавливает права ее владельца, что является важным шагом для защиты интеллектуальной собственности и обеспечения законных интересов разработчика.

#### **4.3.1 Время в течение которого авторские права будут в силе**

Авторские права являются основополагающим аспектом правовой охраны интеллектуальной собственности, обеспечивая авторам контроль над использованием и распространением их творческих произведений. В Российской Федерации регулирование авторских прав осуществляется Гражданским кодексом РФ, где определены основные принципы и сроки действия авторских прав.

Срок действия авторских прав:

- В рамках действующего законодательства Российской Федерации, регулирующего вопросы интеллектуальной собственности, установлены конкретные сроки действия авторских прав. Согласно статье 1286 Гражданского кодекса РФ, права автора на произведение сохраняются на протяжении всей жизни и продолжают действовать в течение семидесяти лет после его кончины. Это обеспечивает долгосрочную защиту творческого наследия и материальные поступления правообладателям.
- В случае создания произведения коллективом авторов, сроки авторских прав определяются исходя из времени жизни каждого

из авторов. По истечении жизни последнего из авторов, права продолжают действовать в течение семидесяти лет, что закреплено в статье 1287 ГК РФ. Это правило гарантирует, что вклад каждого участника коллективного творчества будет уважаем и защищен законом.

- Для произведений, выпущенных анонимно или под псевдонимом, установлен срок действия авторских прав в семьдесят лет с момента первого законного обнародования, как указано в статье 1288 ГК РФ. Это позволяет защитить права автора, даже если его личность остается неизвестной широкой публике.
- Отдельное внимание уделяется аудиовизуальным произведениям, включая фильмы и видеоматериалы. Согласно статье 1301 ГК РФ, авторские права на такие произведения также защищены в течение семидесяти лет после первого обнародования. Это подчеркивает значимость и вклад аудиовизуальных творений в культурное наследие.
- Компьютерные программы, являющиеся результатом интеллектуальной деятельности и технического творчества, также находятся под защитой авторского права. Срок действия прав на компьютерные программы ограничен семидесятью годами после смерти последнего автора, что закреплено в статье 1295 ГК РФ. Это обеспечивает защиту интеллектуальных усилий разработчиков программного обеспечения.

Таким образом, законодательство РФ предусматривает комплексные меры по защите авторских прав, обеспечивая тем самым уважение к творческому труду и интеллектуальной собственности на протяжении значительного времени после смерти автора. Эти нормы способствуют стимулированию культурного и технологического развития общества.

#### **4.3.2 Для оформления свидетельства необходимо выполнить следующие условия:**

Регистрация программного обеспечения как объекта интеллектуальной собственности включает подготовку документации, отражающей уникальные особенности продукта, анализ рынка для подтверждения новизны и избежания нарушения чужих прав. Затем следует подача заявки в Роспатент и экспертиза, в ходе которой проверяется соответствие продукта законодательным требованиям и его инновационный потенциал. В случае удовлетворения всех условий, разработчик получает свидетельство, подтверждающее его права на использование и распространение программного продукта, что способствует стимулированию инноваций и развитию информационного общества. Это обеспечивает защиту интеллектуальной собственности и поддержание правового порядка в сфере информационных технологий.

1. Подготовка документации: Сначала необходимо подготовить пакет документов для подачи заявки на свидетельство. Это включает в себя описание программного продукта, включая его архитектуру, алгоритмы и функциональные особенности, а также схемы, диаграммы и другие материалы, которые могут помочь в понимании технических решений, реализованных в программе.
2. Поиск аналогов: Перед подачей заявки на свидетельство необходимо провести поиск аналогичных программных решений, чтобы убедиться в новизне и оригинальности вашего продукта. Этот этап помогает избежать возможных конфликтов с существующими свидетельствами или авторскими правами.
3. Подача заявки на свидетельство: После подготовки всех необходимых документов и проведения поиска аналогов можно подать заявку на свидетельство в Роспатент. Заявка должна содержать полное описание изобретения, а также утверждение прав на изобретение.
4. Экспертиза заявки: После подачи заявки на свидетельство ее будет рассмотрено экспертами Роспатента. Экспертиза может занять

значительное время и включает в себя проверку соответствия заявки требованиям патентного законодательства, а также ее новизны, промышленной применимости и изобретательского уровня.

5. Внесение поправок и ответ на запросы экспертизы: В случае необходимости могут потребоваться дополнительные материалы или уточнения изначально предоставленных документов. Разработчик должен предоставить ответ на запросы экспертизы и внести все необходимые поправки в заявку.
6. Получение свидетельства: После успешного завершения экспертизы заявка на свидетельство может быть утверждена, и разработчик получит официальное уведомление о выдаче свидетельства. После этого свидетельство будет зарегистрировано в соответствующем реестре, и его владелец получит права на эксклюзивное использование и распространение программного продукта.

В контексте коллективной разработки программного обеспечения, особое внимание уделяется правовому регулированию соавторства. Процесс создания программного кода представляет собой сложное взаимодействие между различными специалистами, каждый из которых вносит уникальный вклад в конечный продукт. В соответствии с положениями статьи 1258 Гражданского кодекса Российской Федерации, признание лица соавтором обусловлено его существенным творческим вкладом в создание программного кода. Это означает, что каждый участник, чей вклад можно чётко идентифицировать в конечном продукте, имеет право на равное участие в правах на результаты интеллектуальной деятельности. Такие права включают в себя не только возможность получения вознаграждения за использование программного продукта, но и признание авторства, что является важным аспектом профессионального признания и репутации.

Тем не менее, важно различать творческий вклад и другие виды деятельности, которые несмотря на свою значимость, не признаются

созданием авторского произведения. Так, специалисты, занимающиеся технической поддержкой, управлением проектами и тестированием, несмотря на важность их работы для успешного функционирования и развития программного продукта, не могут претендовать на статус соавторов, если их деятельность не связана с непосредственным созданием программного кода. Это разграничение имеет ключевое значение для определения круга лиц, имеющих право на авторские права, и предотвращения конфликтов интересов внутри коллективов разработчиков.

Нарушение авторских прав, охраняемых статьёй 1252 Гражданского кодекса РФ, может привести к серьёзным юридическим последствиям. Незаконное использование программного продукта без согласия правообладателя может повлечь за собой гражданско-правовую ответственность, в том числе обязательство возместить ущерб, причинённый правообладателю. В соответствии со статьёй 1301 ГК РФ, размер возмещения ущерба определяется исходя из множества факторов, включая степень вины нарушителя и объём причинённого ущерба. Кроме того, нарушитель может быть обязан компенсировать доходы, полученные в результате незаконного использования программного продукта, что отражено в той же статье и направлено на предотвращение неосновательного обогащения за счёт правообладателя.

Уголовная ответственность за нарушение авторских прав, предусмотренная статьёй 146 Уголовного кодекса РФ, может наступить в случае совершения особо тяжких нарушений. В зависимости от характера и масштаба нарушения, лицо, признанное виновным, может столкнуться с уголовным наказанием, включая лишение свободы на срок до пяти лет. Это подчёркивает серьёзность, с которой российское законодательство относится к защите авторских прав.

Административная ответственность, регламентируемая статьями 7.12 и 14.33 Кодекса РФ об административных правонарушениях, также предусматривает штрафы для лиц, нарушивших авторские права. Размер



штрафов варьируется в зависимости от статуса нарушителя и может быть особенно значительным для крупных организаций, достигая миллионов рублей.

В заключение, соблюдение авторских прав и норм интеллектуальной собственности является неотъемлемым условием для стимулирования инноваций и развития информационного общества. Недооценка этого аспекта может привести к значительным юридическим и финансовым последствиям, подчёркивая важность тщательного учёта вклада каждого участника в процессе разработки и строгого соблюдения законодательства в данной области.

## **ЗАКЛЮЧЕНИЕ**

В ходе преддипломной практики и разработки выпускной квалификационной работы (ВКР) было создано веб-приложение, основанное на фреймворке Django. Данное приложение предоставляет возможность задавать имя и начальные параметры для сканирования целевого домена, что представляет собой важный этап в анализе безопасности сети. В ходе работы над проектом использовались разнообразные инструменты с открытым исходным кодом, предназначенные для сканирования периметра целевой сети и автоматизации процесса. Управление этими инструментами осуществлялось посредством сценариев, написанных на языках `bash` и `Python`.

Процесс сканирования периметра целевой сети был реализован с применением Open Source инструментов, что способствовало повышению гибкости и эффективности при выполнении задач. Скрипты на языке `Python` играли ключевую роль в автоматизации процесса, обеспечивая совместимость приложения с различными операционными системами и возможностями его использования в различных сценариях.

В результате проделанной работы было разработано функциональное веб-приложение, позволяющее эффективно проводить сканирование целевого домена, учитывая предварительно заданные параметры. Данное приложение стало надежным инструментом для анализа безопасности сети, позволяя выявлять потенциальные уязвимости и проводить их анализ.

Использование фреймворка Django предоставило разработке удобство и надежность, позволив сосредоточиться на создании функционала приложения, не отвлекаясь на базовые аспекты веб-разработки. Благодаря этому подходу, время разработки было значительно сокращено, а конечный продукт получил высокое качество.

В целом, итогом проделанной работы стало создание веб-приложения, основанного на фреймворке Django, которое является важным инструментом для анализа безопасности сети и обеспечивает эффективное управление процессом сканирования периметра целевой сети.

Разработанное приложение обладает потенциалом для дальнейшего развития, расширения и применения в различных сферах безопасности информационных систем. В нём присутствует возможность встраивания новых OpenSource инструментов, а также есть возможность переноса полученного web приложения в другие web приложения Django, при необходимости.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. GitHub страница приложения [Электронный ресурс]. URL: <https://github.com/JackoPrograms/skaner/tree/main>
2. Статья – Уязвимости периметра корпоративных сетей // Уязвимости периметра корпоративных сетей [Электронный ресурс] URL: <https://www.ptsecurity.com/ru-ru/research/analytics/vulnerabilities-corporate-networks-2020/#id2> (дата обращения: 21.11.22).
3. Статья 2021 года – Ущерб киберпреступлений в России // Киберпреступления в России [Электронный ресурс] KURL: [https://ria.ru/20211222/kiberprestupleniya\\_1764832102.html](https://ria.ru/20211222/kiberprestupleniya_1764832102.html) (дата обращения: 21.11.22).
4. Справочное руководство по Nmap // Nmap [Электронный ресурс] URL: <https://nmap.org/book/man.html> (дата обращения: 10.11.22).
5. Статья – актуальные киберугрозы // актуальные киберугрозы [Электронный ресурс] URL: <https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2022-q1/> (дата обращения: 02.12.22).
6. Статья – статистика киберпреступлений 2022 – защита от скликивания рекламы // [Электронный ресурс] URL: <https://clickfraud.ru/statistika-kiberprestuplenij-2022/> (дата обращения: 02.12.22).
7. Статистика как перевернулся ИБ за три года // Как перевернулся ИБ за три года [Электронный ресурс] URL: <https://habr.com/ru/company/searchinform/blog/700500/> (дата обращения: 02.12.22)
8. Экономика взломного периода – газета Коммерсант № 234 // Экономика взломного периода [Электронный ресурс] URL: <https://www.kommersant.ru/doc/5722189> (дата обращения: 02.12.22).
9. Статья - Затраты государства на кибербезопасность // Затраты государства на кибербезопасность [Электронный ресурс] URL:

<https://www.cnews.ru/articles/2021-12->

[20\\_gosudarstvo\\_potratit\\_na\\_kiberbezopasnost](#) (дата обращения: 10.12.22).

10. Документация SubFinder // SubFinder – Инструменты Kali Linux URL: <https://kali.tools/?p=4704> (дата обращения: 07.11.22).

11. Документация Subfinder // Subfinder - инструмент обнаружения поддоменов URL: <https://github.com/projectdiscovery/subfinder> (дата обращения: 07.11.22).

12. Документация по использованию инструментов Kali Linux// Kali Linux URL: <https://github.com/ffuf/ffuf> (дата обращения: 10.11.22).

13. Руководство по FFUF// FFUF URL: <https://codingo.io/tools/ffuf/bounty/2020/09/17/everything-you-need-to-know-about-ffuf.html> (дата обращения: 10.11.22).

14. Документация по сканеру безопасности Network Mapper // Network Mapper [Электронный ресурс] URL: <https://nmap.org> (дата обращения: 10.11.22).

15. Справочное руководство по Nmap // Nmap [Электронный ресурс] URL: <https://nmap.org/book/man.html> (дата обращения: 10.11.22).

16. Статья – А что такое API и почему его называют интерфейсом// API [Электронный ресурс] URL: <https://www.calltouch.ru/blog/api-chto-eto-takoe-v-programmirovanii-i-kak-rabotat-s-nim/> (дата обращения: 20.12.22)

17. [Электронный ресурс] //URL: <https://kali.tools/?p=1317> (дата обращения: 10.11.22).

18. DSL на основе YAML – Установка ядер, документация // DSL [Электронный ресурс] URL: <https://github.com/projectdiscovery/nuclei> (дата обращения: 21.11.22)

19. Полное руководство по поиску ошибок с ядрами // The Ultimate Guide to Finding Bugs With Nuclei [Электронный ресурс] URL: <https://blog.projectdiscovery.io/ultimate-nuclei-guide/> (дата обращения: 21.11.22).

20. Документация bash [Электронный ресурс] URL: <https://www.gnu.org/software/bash/manual/bash.html#Bash-Variables>
21. URL: <https://learn.microsoft.com/ru-ru/windows-server/administration/windows-commands/nslookup> — документация nslookup
22. URL: <https://github.com/ffuf> — документация ffuf
23. URL: <https://nmap.org/man/ru/index.html> — документация nmap
24. URL: <https://docs.projectdiscovery.io/introduction> — документация nuclei
25. Страница с шаблонами. Страница проекта [Электронный ресурс] URL: <https://github.com/projectdiscovery/nuclei-templates>
26. Документация sqlite [Электронный ресурс] URL: <https://sqlite.org/docs.html>
27. URL: <https://www.python.org/doc/> – документация python
28. 5 издание книги “Изучаем python” М.Лутц. Том 1
29. 5 издание книги “Изучаем python” М.Лутц. Том 2
30. Отчёт обзора литературы по теме данной работы, Занина Д.С.
31. Базовый образ kalilinux/kali-rolling [Электронный ресурс] URL: <https://hub.docker.com/r/kalilinux/kali-rolling>
32. Руководство по использованию OWASP Amass // Как использовать OWASP Amass: подробное руководство - Dionach URL: <https://www.dionach.com/blog/how-to-use-owasp-amass-an-extensive-tutorial/> (дата обращения: 02.11.22).
33. Документация по OWASP Amass // OWASP Amass – Обширный источник URL: <https://github.com/OWASP/Amass/blob/master/doc/tutorial.md> (дата обращения: 02.11.22).
34. Документация Assetfinder // Assetfinder – Поиск доменов и поддоменов, связанных с данным доменом URL: <https://www.geeksforgeeks.org/assetfinderfind-domains-and-subdomains-related-to-a-given-domain/> (дата обращения: 07.11.22).

35. Документация по Web Application Pentest // Subdomain Enumeration Tools [Web Application Pentest] URL: [https://www.golinuxcloud.com/subdomainenumeration-tools/#2\\_Assetfinder](https://www.golinuxcloud.com/subdomainenumeration-tools/#2_Assetfinder) (дата обращения: 07.11.22).

36. Сканер веб путей dirsearch документация // Dirsearch URL: <https://github.com/maurosoria/dirsearch> (дата обращения: 10.11.22)