**Федеральное государственное автономное образовательное учреждение высшего образования**

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Факультет: Факультет информационных технологий  
Кафедра «Информационная безопасность»**

**Направление подготовки/специальность: 10.03.01 Информационная безопасность**

**ОТЧЕТ**

**по проектной практике**

**Студент: Зверев Захар Дмитриевич Группа: 241-351**

**Место прохождения практики: Московский Политех, кафедра Информационная безопасность**

**Отчет принят с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Руководитель практики: Гнешев А. Ю., к.т.н., доцент кафедры «Информационная безопасность»**

**Москва 2025**

Оглавление

[ОБЩАЯ ИНФОРМАЦИЯ О ПРОЕКТЕ 3](#_Toc198589083)

[Общая информация о проекте 3](#_Toc198589084)

[Общая характеристика деятельности организации 3](#_Toc198589085)

[ОПИСАНИЕ ДОСТИГНУТЫХ РЕЗУЛЬТАТОВ ПО ПРОЕКТНОЙ ПРАКТИКЕ 6](#_Toc198589086)

[ИЗУЧЕНИЕ MITRE ATT&CK 8](#_Toc198589087)

[ИЗУЧЕНИЕ OWASP 10](#_Toc198589088)

[РАЗБОР ИНЦЕДЕНТА 13](#_Toc198589089)

[Настройка Git и репозитория 15](#_Toc198589090)

[Написание документов в Markdown 15](#_Toc198589091)

[Создание статического веб-сайта 15](#_Toc198589092)

[Взаимодействие с организацией-партнёром 16](#_Toc198589093)

[Итоговые навыки и достижения 16](#_Toc198589094)

[ВАРИАТИВНАЯ ЧАСТЬ 16](#_Toc198589095)

[Static application security testing (SAST) 17](#_Toc198589096)

[Dynamic application security testing (DAST) 18](#_Toc198589097)

[Software Composition Analysis (SCA) 20](#_Toc198589098)

[Фаззинг (Fuzz Testing) 21](#_Toc198589099)

[Источники: 23](#_Toc198589100)

# ОБЩАЯ ИНФОРМАЦИЯ О ПРОЕКТЕ

Данный отчет посвящен описанию опыта, приобретенного в ходе учебной (проектной) практики, проходившей с 03 февраля 2025 г. по 24 мая 2025 г. по в Московском Политехническом университете. Целью практики являлось закрепление теоретических знаний, полученных в процессе обучения, приобретение практических навыков в области информационной безопасности, веб-разработки, анализа данных, а также ознакомление с реальными задачами, возникающими в профессиональной деятельности.

## Общая информация о проекте

Название проекта: «Команда CTF».

Capture The Flag (CTF) — это соревнования в области информационной

безопасности, где участники решают различные задачи для получения "флагов". Флаг — это секретная строка, которая подтверждает успешное решение задачи. Целью CTF является развитие практических навыков в области информационной безопасности и подготовка к реальным задачам.

Цель: Создание сообщества для тренировки студентов в области кибербезопасности, предоставляющего комплексное обучение как атакующим, так и защитным методам, подготовку профессиональной команды CTF, а также создание платформы для проведения соревнований и других образовательных мероприятий. А также создание устойчивой команды для участия в CTF соревнованиях из студентов ФИТа 1 курса.

## Общая характеристика деятельности организации

Наименование заказчика: ООО «НЛБ».

Компания разделена на направление отделы и группы: группа разработки, группа технической поддержки, ситуационный центр, отдел продаж, руководство.

Проектирование и установка систем безопасности и автоматизации, сервисное обслуживание, комплексные поставки оборудования, предотвращение потерь в торговле и на производстве - основные направления деятельности Группы Компаний «Национальные Лаборатории Безопасности».

Среди заказчиков компании множество крупных коммерческих и государственных организаций. Группа Компаний участвует в целевых программах правительства Москвы в области обеспечения безопасности как отдельных объектов, так и целых районов.

Выполнение проектов осуществляется с учетом всех современных норм и требований. Специалистами компании обеспечивается качественный монтаж, пусконаладочные работы, сдача всех систем «под ключ», гарантийное и сервисное обслуживание, а также оперативность и конфиденциальность работ.

**ОПИСАНИЕ ЗАДАНИЯ ПО ПРОЕКТНОЙ ПРАКТИКЕ**

В рамках проектной практики было поставлено комплексное задание, направленное на формирование навыков работы с современными инструментами разработки, управления проектами и взаимодействия с организациями-партнёрами. Задание разделено на базовую и вариативную части, каждая из которых предусматривала выполнение конкретных задач.

**Базовая часть** включала настройку системы контроля версий Git с созданием репозитория на платформе GitHub или GitVerse на основе предоставленного шаблона. Требовалось освоить базовые команды: клонирование, коммит, отправку изменений и управление ветками, а также регулярно фиксировать прогресс с осмысленными комментариями. Параллельно необходимо было оформить всю проектную документацию в формате Markdown, изучив его синтаксис для подготовки описаний, журналов прогресса и других материалов.

Ключевым этапом базовой части стало создание статического веб-сайта, посвящённого проекту по дисциплине «Проектная деятельность». Для реализации допускалось использование HTML и CSS, но рекомендовалось применение генератора Hugo для упрощения процесса. Сайт должен был включать домашнюю страницу с аннотацией проекта, разделы «О проекте», «Участники» с описанием личного вклада каждого студента, «Журнал» с тремя записями о прогрессе и «Ресурсы» со ссылками на материалы партнёрской организации. Оформление требовало уникальности более чем на 50%, а также интеграции графических и медиаматериалов.

Важным аспектом стало взаимодействие с организацией-партнёром: участие в профильных мероприятиях (конференциях, семинарах, мастер-классах,

экскурсиях) и организация встреч или стажировок. Итоговый отчёт о проделанной работе, включая описание полученного опыта и знаний, необходимо было оформить в PDF и DOCX форматах и разместить в репозитории и на сайте.

**Вариативная часть** предполагала углубление в тему кибербезопасности: изучить методы анализа защищённости программного обеспечения – инструменты SAST, DAST и др. Также – провести сравнительное описание таких методов.

Работа над заданием базировалась на предыдущих этапах, включавших изучение MITRE ATT&CK для анализа тактик злоумышленников, знакомство с OWASP Top 10 для идентификации уязвимостей веб-приложений, а также разбор реального инцидента информационной безопасности. Это позволило глубже понять контекст проектной практики и применить полученные знания для реализации вариативной части.

# ОПИСАНИЕ ДОСТИГНУТЫХ РЕЗУЛЬТАТОВ ПО ПРОЕКТНОЙ ПРАКТИКЕ

В самом начале практики было выдано общее задание, которое включало в себя следующие пункты: изучение и описание основных аспектов матрицы — Mitre Att&ck; изучение и описание информации с сайта OWASP; разбор реального инцидента произошедшего за последний год-полтора, с требованием расписать какие тактики, техники и процедуры были применены злоумышленниками.

В ходе выполнения задания, посвящённого изучению MITRE ATT&CK, сайту OWASP и анализу реального инцидента, были достигнуты значимые результаты, которые позволили углубить понимание современных угроз информационной безопасности и методов противодействия им. Изучение матрицы MITRE ATT&CK обеспечило систематизацию знаний о тактиках, техниках и процедурах (TTPs), используемых злоумышленниками на различных этапах кибератак. Это позволило не только классифицировать методы атак, такие как lateral movement, credential dumping или execution через вредоносные скрипты, но и научиться прогнозировать возможные векторы угроз в контексте конкретных инфраструктур.

Анализ материалов OWASP, включая актуальную версию OWASP Top-10, дал чёткое представление о наиболее критичных уязвимостях веб-приложений, таких как инъекции, недостаточная защита данных или misconfiguration. Это знание было основой для понимания базовых принципов безопасной разработки. Особое внимание было уделено изучению OWASP Application Security Verification Standard, Mobile ASVS, Web Security Testing Standard и cheat-sheet’ов.

Разбор реального инцидента, произошедшего в 2024–2025 годах, стал практическим применением полученных теоретических знаний. В ходе анализа были идентифицированы тактики атаки по матрице MITRE ATT&CK, такие как Initial Access (через фишинговые письма), Privilege Escalation (использование уязвимости в ПО) и Exfiltration (передача данных через зашифрованные каналы). Для каждой техники были определены соответствующие процедуры, включая инструменты, использованные злоумышленниками. Это позволило не только реконструировать цепочку атаки, но и выделить ключевые точки, где можно было бы предотвратить или обнаружить инцидент на ранних этапах.

Итогом работы стало формирование комплексного отчёта, который можно видеть ниже или в папке task в Git репозитории (примерное время выполнения 4 часа)

# ИЗУЧЕНИЕ MITRE ATT&CK

MITRE – некоммерческая американская организация, занимающаяся исследованиями в области регулирования и навигации в воздушном пространстве, систем глобального позиционирования (GPS), аэрокосмической промышленности, *кибербезопасности* и других.

Кроме разбираемой в этой работе матрицы MITRE ATT&CK, у этой компании есть и другие общедоступные проекты в области кибербезопасности, например каталоги CVE (Common Vulnerabilities and Exposures, общедоступный стандартизированный список уязвимостей) и CWE (Common Weakness Enumeration, список дефектов безопасности программного обеспечения).

«ATT&CK» расшифровывается как «Adversarial Tactics, Techniques and Common Knowledge».

Эта матрица представляет из себя общедоступную базу знаний, основанную на анализе реальных атак, поделённую на этапы. В ней содержится список тактик (заголовки столбцов), а также техник и подтехник (содержимое столбцов) для этой тактики. Нужна матрица MITRE ATT&CK для описания «паттернов поведения злоумышленников» и используется в качестве основы для разработки конкретных моделей угроз и методологий в сфере кибербезопасности.

Есть 3 матрицы MITRE ATT&CK – для корпоративных сетей и классических клиент-сервисных приложений (Enterprise ATT&CK), для мобильных приложений (Mobile ATT&CK) и для промышленных систем управления (ICS ATT&CK).

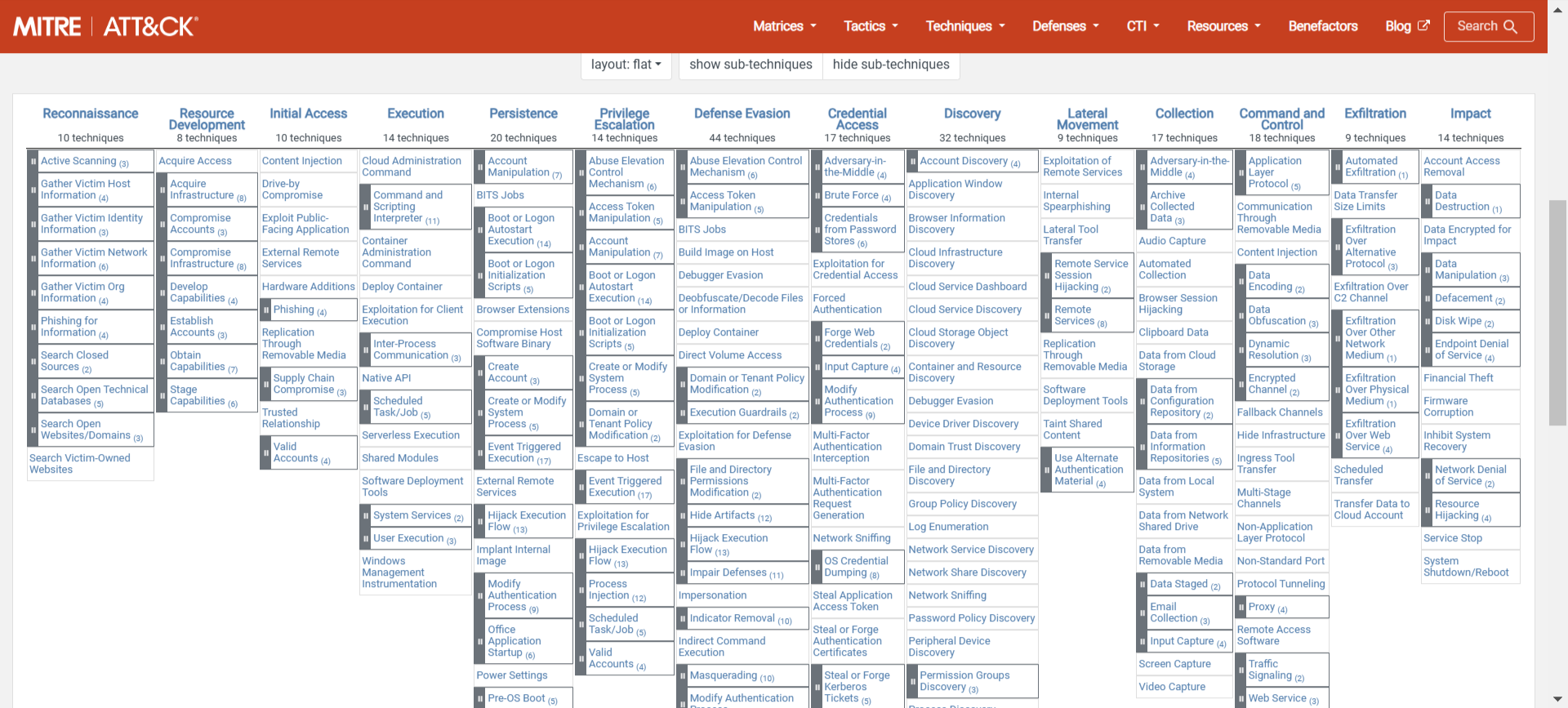


Рис. 1, неполная матрица MITRE ATT&CK

Также, на сайте MITRE ATT&CK, кроме матриц и информации о тактиках, техниках и подтехниках, можно найти информацию о «группировках» («кластерах активности с общими названиями») злоумышленников, а также информацию об их «типичном поведении» (рис. 2 и рис. 3).

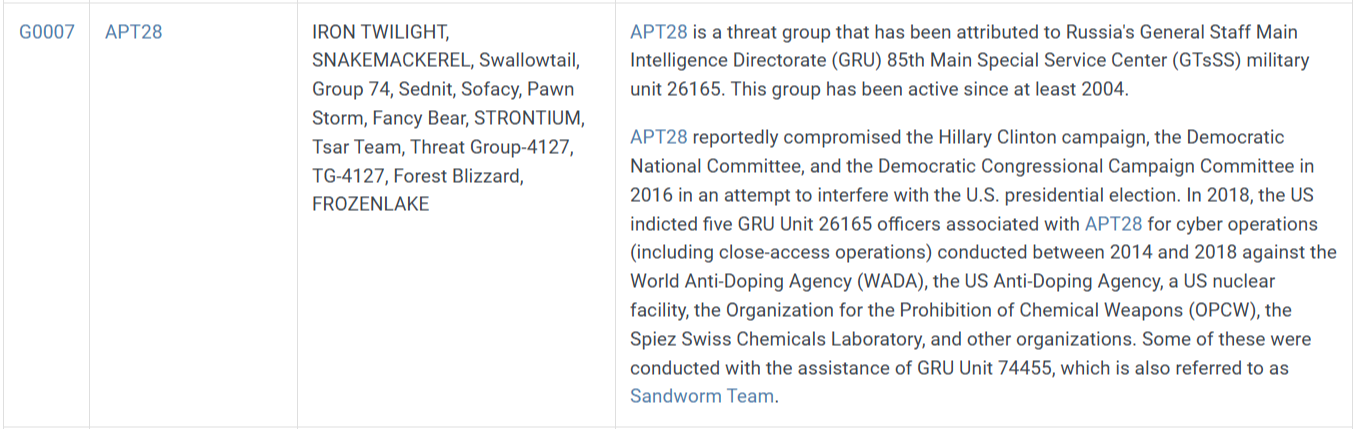


Рис. 2, информация об APT28 в разделе «Groups»

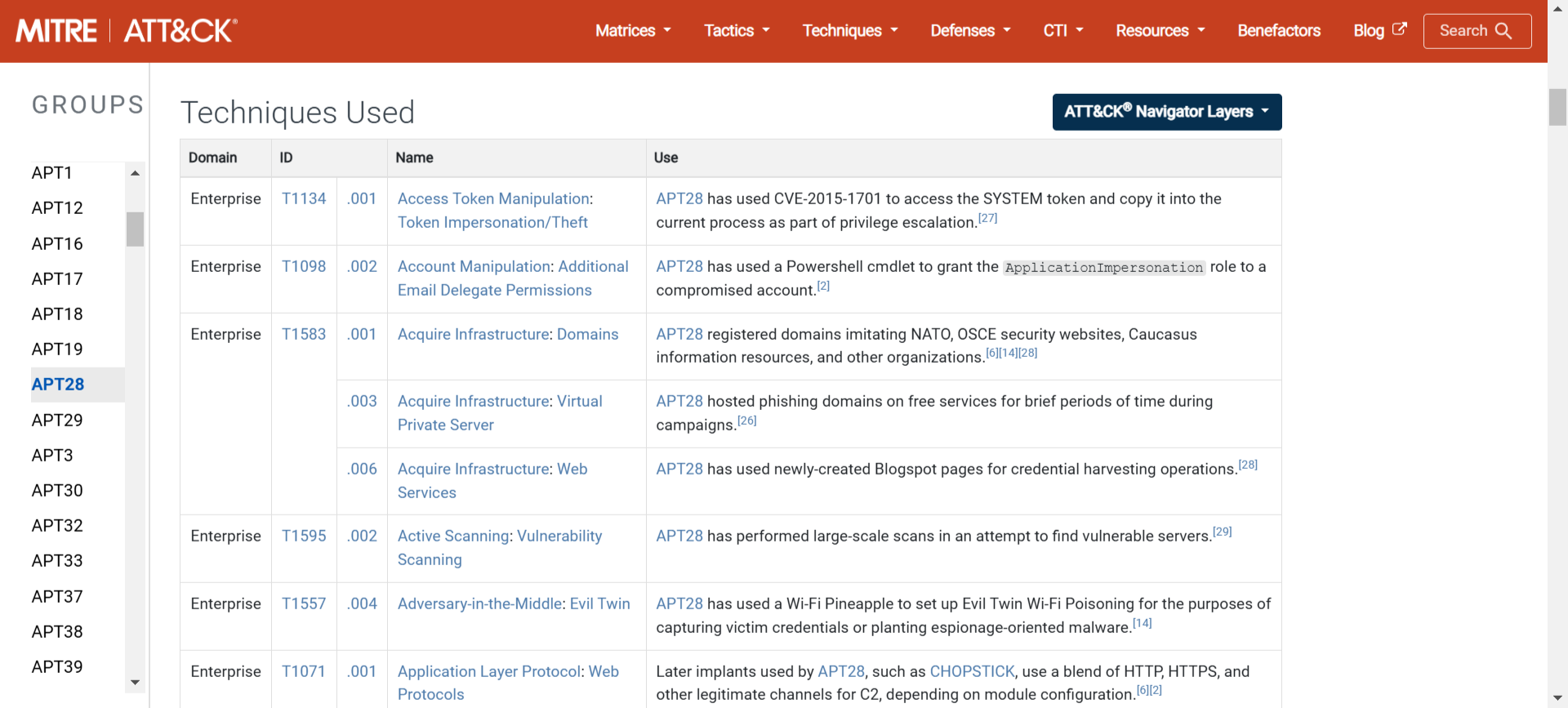


Рис. 3, неполный список используемых APT28 техник

# 

# ИЗУЧЕНИЕ OWASP

OWASP или «Open Worldwide Application Security Project» – это международная некоммерческая организация, ставящая своей целью улучшение безопасности веб-приложений и другого программного обеспечения. Один из основных принципов OWASP заключается в том, что все их материалы являются общедоступными, и их можно найти на их веб-сайте.

Самый известный проект OWASP – это OWASP Top-10. Это обновляющийся раз в несколько лет отчёт с 10 самыми частыми проблемами безопасности (уязвимостями) веб-приложений. На данный момент актуальна версия 2021 года, но в первой половине 2025 года планируется публикация нового отчёта.



Рис. 4, сравнение и изменение OWASP Top-10 2017 и 2021

OWASP ASVS (Application Security Verification Standard) – это проект OWASP, представляющий из себя стандарт для проведения проверок уровня безопасности приложений. Основная цель этого проекта – «нормализовать диапазон охвата и уровень строгости, когда речь идёт о проверке безопасности веб-приложений».

Предполагается ссылка на требования ASVS в установленном формате. Делается это для разных целей: как указание для разработчиков или сторонних лиц, обеспечивающих безопасность приложений.  
Формат ссылки: <chapter>.<section>.<requirement>; на каждой из 3 позиций ставится число.  
Также может указываться версия ASVS, тогда формат будет выглядеть иначе: v<version>-<chapter>.<section>.<requirement>; если версия не указывается, ссылка считается ссылающей на требование в новейшей версии.

OWASP Gen AI – проект по обеспечению безопасности генеративного ИИ. Этот проект включает в себя «OWASP Top-10 LLM and Gen AI» – список 10 основных уязвимостей для языковых моделей и генеративного ИИ. На сайте этого проекта есть документ на русском языке с «Топ-10 OWASP для приложений LLM 2025» от 11 марта 2025 г., с наиболее частыми уязвимостями, объяснением их причин и методами устранения (документ доступен не только на русском).

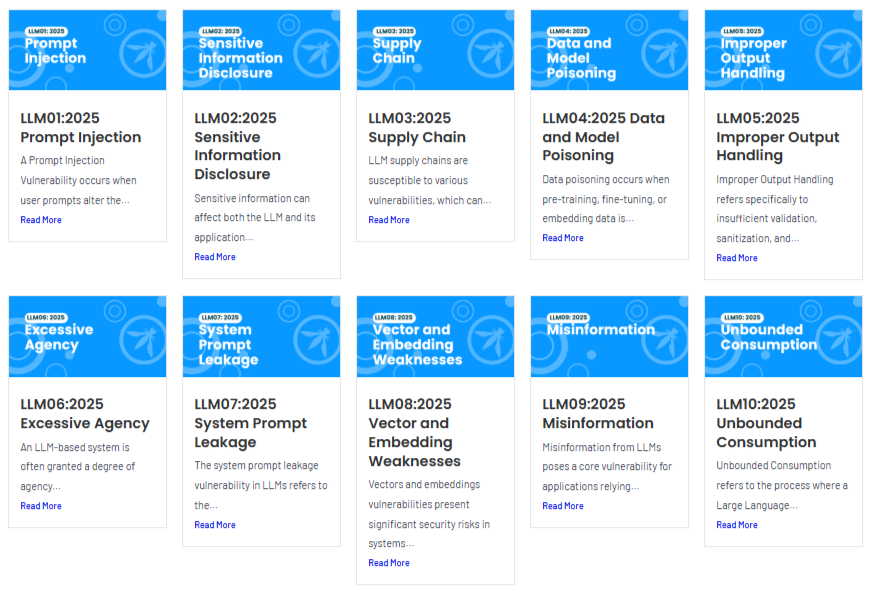


Рис. 5, OWASP Top-10 LLM and Gen AI 2025

OWASP Juice Shop – это «намеренно-небезопасное» веб-приложение. Оно создано для тренингов по безопасности, демонстраций уязвимостей, командных турниров и в качестве «подопытного кролика» для тестирования инструментов безопасности. OWASP Juice Shop содержит уязвимости из OWASP Top-10 и другие. Во время выполнения лабораторных работ на дисциплине «проектная деятельность» я запускал это приложение в контейнере Docker и тестировал на нём инструменты проверки безопасности.

OWASP MAS (Mobile Application Security) – стандарт безопасности для мобильных приложений (MASVS, Mobile ASVS) и подробное руководство по тестированию мобильных приложений (MASTG). В них описываются процессы, методы и инструменты, используемые при тестировании безопасности мобильных приложений, а также исчерпывающий набор тестовых сценариев, позволяющий тестировщикам получать согласованные и полные результаты. Текущая версия MASVS – 2.1.0, MASTG – 1.7.0.

OWASP WSTG (Web Security Testing Guide) - подробное руководство по тестированию безопасности веб-приложений и веб-сервисов. WSTG создан специалистами по кибербезопасности. OWASP называет его «свод лучших практик, используемых тестировщиками на проникновение и организациями по всему миру». Форма ссылки на WSTG: WSTG-<category>-<number>. Форма с указанием версии документа: WSTG-<version>-<category>-<number>.

OWASP ZAP – бесплатный open-source сканер уязвимостей вроде Burp Suite. Позволяет проводить ручное и автоматическое сканирование веб-приложений. OWASP ZAP я также использовал при выполнении лабораторных работ на «проектной деятельности».

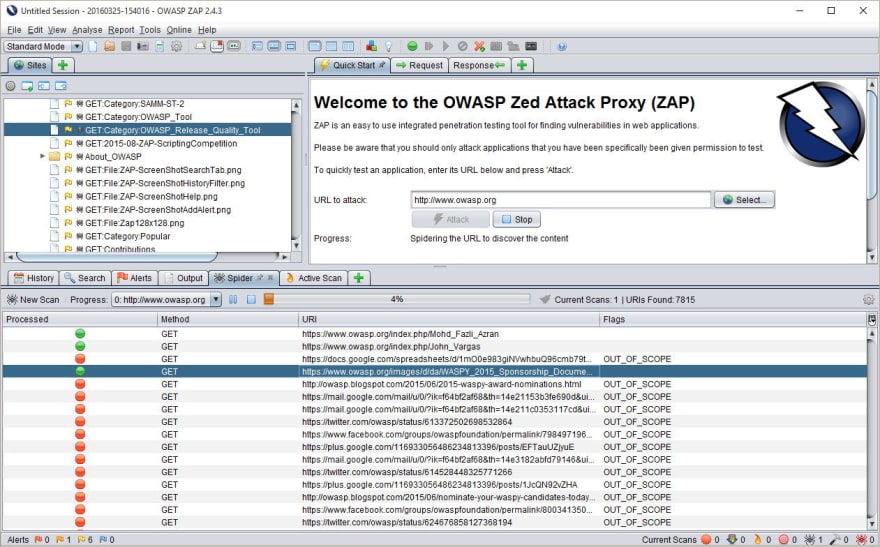


Рис. 6, пример интерфейса OWASP ZAP

OWASP Dependency-Check – это инструмент определения компонентов программного обеспечения (SCA), который пытается обнаружить известные уязвимости зависимостей проекта. Оно было создано после попадания уязвимости «использование компонентов и библиотек с известными уязвимостями» в список OWASP Top-10 2013.

Это далеко не все проекты OWASP – в их GitHub более 1300 репозиториев, а предложить проект может любой. Но перечисленные и описанные выше проекты являются основными и наиболее используемыми.

# РАЗБОР ИНЦЕДЕНТА

В 2024 году было значительное количество атак APT-группы SideWinder в Джибути. Позже злоумышленники переключились на другие организации в Азии, а также проявляли особое внимание к целям в Египте. Кроме того, был зафиксирован ряд атак, которые свидетельствуют об особом интересе группы к атомным электростанциям и ядерной энергетике в Южной Азии, а также о постепенном расширении ее деятельности на новые страны, особенно в Африке.

SideWinder постоянно совершенствует свой инструментарий, чтобы обходить механизмы детектирования защитного ПО, лучше закрепляться в скомпрометированных сетях и скрывать свое присутствие в зараженных системах. На основе наблюдений «Лаборатории Касперского» за деятельностью группы есть предположение, что она постоянно отслеживает случаи детектирования своего инструментария. Сразу после того, как средства безопасности обнаруживают инструменты SideWinder, злоумышленники оперативно – иногда за считаные часы – создают новую или модифицированную версию вредоносного ПО. Если обнаружение произошло на основе поведенческого анализа, APT-группа SideWinder обычно меняет методы закрепления в системе и загрузки компонентов.

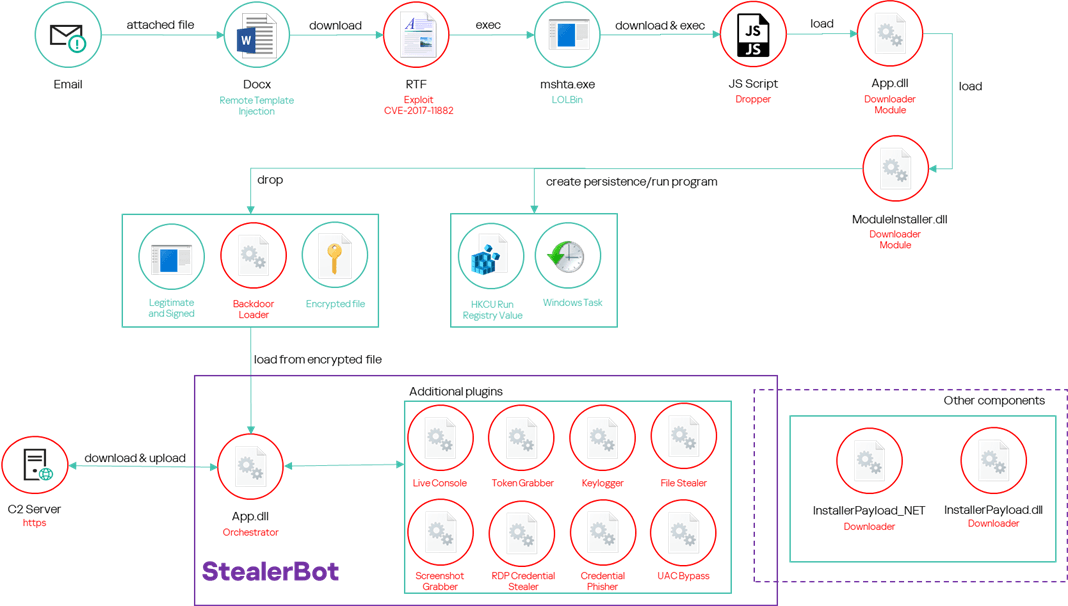


Рис. 7, схема заражения устройств

Злоумышленники рассылают целевые фишинговые письма с прикрепленным файлом DOCX. В этом документе применяется техника инъекции удаленного шаблона для загрузки RTF-файла с удаленного сервера, находящегося под контролем злоумышленников. В файле эксплуатируется известная уязвимость CVE-2017-11882 «Microsoft Office Memory Corruption Vulnerability» для выполнения вредоносного шелл-кода и запуска многоэтапного процесса заражения, в результате которого устанавливается ВПО, названный в «Лаборатории Касперского» Backdoor Loader. Он служит загрузчиком для StealerBot – набора посткомпрометационных инструментов, который используется исключительно группой SideWinder.

В последней волне атак целями группы SideWinder также были дипломатические учреждения в Афганистане, Алжире, Болгарии, Китае, Индии, Руанде, Саудовской Аравии, Турции и Уганде, а также на Мальдивах.

# БАЗОВАЯ ЧАСТЬ

После выполнения общей части была выполнена базовая часть, которая включала в себя следующие пункты: настройка Git и репозитория, написание документов в Markdown, создание статического веб-сайта. Выполнение базовой части задания позволило закрепить ключевые навыки работы с современными инструментами разработки, управления версиями и документацией, а также создать функциональный продукт в виде статического веб-сайта.

# Настройка Git и репозитория

Была успешно организована работа с системой контроля версий Git: создан репозиторий на платформе GitHub, освоены базовые команды, включая клонирование, создание веток, фиксацию изменений с осмысленными комментариями и отправку кода в удалённое хранилище. Регулярные коммиты обеспечили прозрачность истории разработки, а разделение задач через ветки позволило эффективно распределять работу между участниками команды. Репозиторий стал централизованной платформой для хранения всех материалов проектной практики, включая исходный код сайта, документацию и отчёты. (затрачено 4 часа)

# Написание документов в Markdown

Документация проекта была полностью оформлена в формате Markdown, что повысило её структурированность и удобство чтения. Изучен синтаксис для работы с заголовками, списками, таблицами, гиперссылками и вставкой изображений. Созданы такие материалы, как описание проекта, технические спецификации и инструкции. Документы были интегрированы в репозиторий, что обеспечило их доступность для всех участников команды и упростило дальнейшее сопровождение проекта. (затрачено 1 час)

# Создание статического веб-сайта

Разработан статический веб-сайт, посвящённый проектной деятельности. Для разработки сайта было выбрано сочитание языка разметки HTML и CSS. Сайт включает:

* Главную страницу с краткой аннотацией проекта;
* Раздел «О проекте» с детальным описанием целей и задач;
* Страницу «Участники», где указан вклад каждого члена команды;
* «Журнал прогресса» с тремя записями, отражающими ключевые этапы работы;
* Раздел «Ресурсы» со ссылками на материалы организации-партнёра и полезные источники.

Уникальность контента и дизайна была обеспечена за счёт авторских решений: адаптивной вёрстки на HTML/CSS, интеграции графики (фотографий) и медиаэлементов (видео). Сайт размещён в репозитории. (затрачено примерное 8 часов)

# Взаимодействие с организацией-партнёром

В рамках взаимодействия с организацией-партнёром я посетил мастер-класс Инфосистемы Jet (2 часа) и был на экскурсии в офис кампании R-vision (4 часа).

# Итоговые навыки и достижения

* Освоены инструменты DevOps: Git, GitHub, работа с ветками и pull-request.
* Приобретён опыт структурированного документирования в Markdown.
* Развиты навыки фронтенд-разработки, включая вёрстку, работу с HTML и CSS и публикацию исходного кода проекта.

Результаты базовой части стали фундаментом для реализации вариативной задачи, связанной с настройкой WAF, а также продемонстрировали способность работать в команде, соблюдать сроки и применять современные ИТ-инструменты на практике.

# ВАРИАТИВНАЯ ЧАСТЬ

**"Сравнительное описание методов анализа защищенности ПО"**

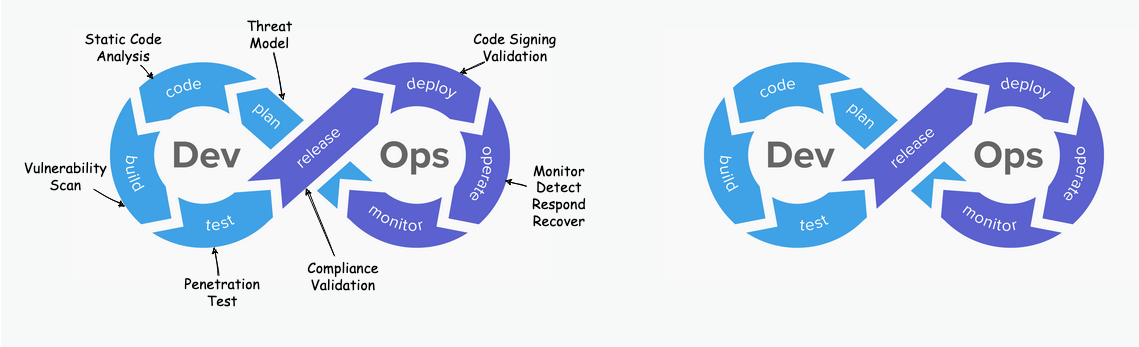
****

Рис. 8, применение решений для анализа защищённости ПО на разных этапах DevSecOps

В современных веб-приложениях безопасность критична: по статистике около 90% инцидентов связаны с эксплуатацией известных ошибок в ПО. Поэтому защита должна реализовываться по многоуровневой модели, совмещающей различные методы анализа. Ключевые методики включают статический анализ исходного кода (SAST), динамический анализ (DAST), анализ состава ПО (SCA) и фаззинг-тестирование. Все они применяются на разных этапах жизненного цикла и дополняют друг друга.

## Static application security testing (SAST)

**Принцип действия:** Анализ по принципу «white box» - проверка кода без его выполнения. Инструменты SAST разбирают исходный код (или скомпилированный код), ищут в нем известные паттерны опасного поведения и нарушения правил безопасного программирования. Например, анализируются потоки данных ("data flow"), шаблоны передачи параметров, использование API и конструкций (регулярные выражения, SQL-запросы и др.) для выявления уязвимостей. Поскольку код не исполняется, SAST может применяться на ранних этапах разработки, уже при написании или проверке исходников. **Этап жизненного цикла:** Чаще всего SAST выполняют на этапе разработки. Инструменты SAST можно интегрировать в систему контроля версий (проверка кода SAST'ом может происходить автоматически при "коммите" изменений в системе контроля версий) или в "CI-пайплайн" . Такое размещение позволяет сразу обнаруживать и устранять дефекты безопасности до релиза. **Достоинства:**

* Раннее обнаружение уязвимостей – на этапе разработки ПО, до запуска приложения. Это снижает время обнаружения ошибок и повышает безопасность.
* Обнаружение хорошо известных уязвимостей на уровне кода: SQL-инъекций, Remote Code Execution, переполнений буфера и т.п. Инструмент точно указывает файл и строку, где найден опасный фрагмент.
* Помогает соответствовать стандартам кодирования и требованиям (например, OWASP ASVS или OWASP MASVS). **Ограничения:**
* SAST не видит поведение программы во время выполнения, поэтому пропускает ошибки конфигурации, проблемы аутентификации/авторизации, "уязвимости бизнес-логики" и другие "динамические дефекты".
* Частые ложные срабатывания. Статический анализ часто выдает подозрительные фрагменты, которые требуют ручной проверки. Нередко бывает много ложных срабатываний и пропусков из-за ограниченности правил выявления ошибок.
* Требует анализируемого и компилируемого кода, т. е. если проект не собирается или сильно зависит от внешних данных, анализаторы могут не видеть ошибок в них.
* **Ограничен известными шаблонами уязвимостей (!).** SAST не обнаружит неизвестные (0-day) ошибки в новом коде и, как правило, не справится с анализом динамически генерируемых строк. **Примеры инструментов.** Популярные SAST-сканеры: SonarQube (Open source приложение, работает с различными языками), CodeQL (Open source, мультиязычный), Semgrep (мультиязычный; бесплатен для команд менее 10 человек), Bandit (Бесплатный, работает только с Python) и другие.

## Dynamic application security testing (DAST)

**Принцип действия.** DAST – это тестирование по принципу «черного ящика» на запущенном приложении. Инструменты DAST (обычно это сканеры веб-приложений) моделируют атаки извне: они запускают приложение (чаще всего в тестовой среде) и автоматически осуществляют прокси-сканирование, посылают запросы с «вредоносными» нагрузками, пытаясь обнаружить уязвимости на живой системе. То есть DAST работает как автоматизированное тестирование на проникновение (penetration test): мониторит трафик между клиентом и сервером, выясняет структуру приложения ("crawling") и пытается ввести вредоносные данные (например, SQL-инъекции, вредоносные скрипты для XSS и др.) во все формы, параметры или заголовки. **Этап жизненного цикла.** DAST проводится после сборки и запуска приложения: на этапе тестирования и даже иногда после выпуска. Инструмент должен иметь доступ к развёрнутому приложению. В отличие от SAST, DAST не требует исходного кода – достаточно запущенного приложения. Весь анализ происходит на этапе тестирования, «готового» продукта. **Достоинства.** Главные плюсы DAST:

* Обнаружение уязвимостей в **работающем приложении**. Так можно найти ошибки, вызванные интеграцией компонентов или специфическими конфигурациями, которые статический анализ не видит.
* Невозможность скрыть дефекты исходным кодом – сканер проверяет реальные ответы сервера. К тому же некоторые DAST'ы способны «замечать» серверные ошибки, расшифрованные данные, подбирать скрытые конечные точки атаки и даже расшифровывать ответы.
* Не требуется исходный код – можно применять к любому запущенному веб-приложению.
* Автоматизация поиска "пост-релизных проблем" – решения DAST можно запускать по расписанию для поиска ошибок. **Ограничения.** К недостаткам DAST относятся:
* Требуется доступ к развернутому приложению. Инструмент должен сканировать URL и выполнять HTTP-запросы, поэтому он не может протестировать код, который недоступен без прямого подключения.
* Ограниченность покрытия: если приложение сложное (много шагов аутентификации, специфических сценариев), сканер может не достичь всех участков. Для этого нужны дополнительные настройки (например, скрипты логирования и т. п.).
* Менее точное локализованное указание на код: DAST найдет уязвимость по URL и параметру HTTP запроса, но не укажет источник ошибки. Часто приходится смотреть логи и код приложения.
* Возможные ложные срабатывания и риски воздействия на запущенное приложение (сканер может затронуть функциональность или вызвать дополнительную нагрузку). **Примеры инструментов:** OWASP ZAP (Open source), Burp Suite (есть бесплатная версия), Nikto ("командный сканер HTTP") и другие.

## Software Composition Analysis (SCA)

**Принцип действия.** SCA – анализ сторонних компонентов и зависимостей приложения. Инструмент SCA сканирует файлы зависимостей (например, package.json, requirements.txt, composer.lock) или просматривает код, чтобы выявить **все библиотеки и пакеты со сторонним исходным кодом.** Затем найденные компоненты соотносятся с базами уязвимостей (например CVE или CWE) и источниками. Цель – обнаружить известные уязвимости и проблемы лицензирования во включенных сторонних модулях. **Этап жизненного цикла.** SCA обычно выполняется уже на ранних стадиях разработки и сборки при добавлении новых зависимостей. Таким образом, еще до релиза можно заметить, что пакет с открытым исходным кодом содержит CVE или CWE (или создан "недоверенным" источником). Часто SCA-инструменты работают непрерывно, поддерживая актуальный **SBOM (Software Bill of Materials)** всего приложения. **Достоинства.** SCA позволяет:

* Быстро обнаружить **известные уязвимости в сторонних библиотеках**, не начиная с исходного кода. Инструмент сравнивает версии с базами данных (например, NVD).
* Контролировать лицензирование и обновления компонентов. Так можно вовремя обновить уязвимые пакеты или отказаться от небезопасных.
* Формировать SBOM: полный перечень зависимостей для аудита и соответствия стандартам. Многие SCA-решения генерируют SBOM автоматически.
* Иметь относительно небольшое число ложных срабатываний, так как поиск ведётся по записям CVE. **Ограничения.** Минусы SCA:
* Находит **только известные уязвимости**. Новые (zero-day) и ошибки, специфичные для конкретного приложения (в его коде), он не обнаружит.
* Зависит от актуальности баз данных: если пакет недавно получил CVE, нужно обновить базы.
* Множество зависимостей (или зависимости от зависимостей) усложняют анализ: иногда нужно просматривать "глубоко вложенные" пакеты.
* Не ищет уязвимости в собственном коде приложения, только в библиотеках, то есть требует сочетания с SAST/DAST. **Примеры инструментов:** OWASP Dependency-Check (Open source), Dependency-Track, Chekov (Open source) и другие.

## Фаззинг (Fuzz Testing)

**Принцип действия.** Фаззинг – это динамический метод тестирования, основанный на подаче не-валидных, искажённых, случайных или "специализированных" (вредоносных) входных данных с целью вызывать сбои или неожиданные поведения приложения. Фаззер (fuzzer) генерирует тестовые кейсы («случайные» строки, последовательности байт, входы HTTP-запросов и т.д.), подаёт их в приложение и отслеживает сбои: падения производительности, исключения, утечки памяти, тайм-ауты и иные «аномальные» реакции. Ключевые компоненты фаззинга – генератор входных данных, их передача в целевое ПО и "detector", который фиксирует отклонение (например, прекращение работы приложения или его компонента после определённого ввода). **Этап жизненного цикла.** Фаззинг применяется на этапе тестирования продукта: после того, как приложение собрано и доступно для пост-релизного тестирования. В отличие от DAST, фаззинг может выполняться даже после релиза, а не только в тестовой среде, для долгосрочного поиска редких ошибок.

**Достоинства:**

* **Поиск неизвестных ошибок.** Он не ищет конкретную уязвимость, а выявляет любые входы, приводящие к сбоям. Таким образом можно обнаружить очень неочевидные баги, включая «0-day», которые стандартные анализаторы "не знают".
* **Низкий процент ложных срабатываний.** Продвинутые "фазз-тестеры" дают подтверждённые эксплойты для найденных уязвимостей – проверенное падение или сбой – а не "подозрение". Это значительно упрощает верификацию найденных уязвимостей.
* **Критерии охвата.** Фаззинг продолжает генерировать тесты даже после обнаружения первой проблемы, постепенно охватывая новые варианты запросов. Современные технологии ("greybox-фаззинг с измерением покрытия") повышают шанс найти серьезные ошибки.
* **Обнаружение специфических дефектов:** подходит для поиска ошибок, связанных с недостаточной проверкой ввода: переполнений буфера, "race condition", некорректной обработки бинарных/структурированных данных (файловые форматы, JSON, XML и т.д.) и прочего. **Ограничения.** Недостатки фаззинга:
* Требует доступа к интерфейсу ввода данных. Например, фаззингом сложно охватить код, не используемый входом для данных.
* Может пропустить уязвимости, требующие сложных условий, не достижимых случайными данными.
* Зависит от времени и ресурсов: чтобы глубоко протестировать большие приложения, нужны долгие сеансы или распределённый фаззинг.
* Меньше подходит для поиска “бизнес-логики” и проблем, не приводящих к сбою или сбою модели безопасности.
* Подходит не для всех языков/платформ (например, интерпретируемые языки сложнее «фаззить» через прямые бинарные ошибки). **Примеры инструментов:** Honggfuzz, Peach Fuzzer, Radamsa, OWASP ZAP Fuzzer, Burp Intruder и другие.

## Источники:

<https://www.anti-malware.ru/analytics/Technology_Analysis/SAST-DAST-SCA-SCS-development-security>

<https://forwardsecurity.com/sast-sca-dast-iast-rasp-what-they-are-and-how-you-can-automate-application-security/#:~:text=RASP%20agents%2C%20like%20IAST%20tools%2C,to%20block%20such%20entry%20automatically>

<https://owasp.org/www-project-devsecops-guideline/latest/02a-Static-Application-Security-Testing#:~:text=Static%20Code%20Analysis%20or%20Source,code%20without%20executing%20the%20program>

<https://owasp.org/www-project-web-security-testing-guide/>

<https://rt-solar.ru/products/solar_appscreener/blog/4446/>

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проектной практики был реализован комплексный проект, сочетающий исследовательскую и техническую составляющие. Основной акцент работы был сделан на изучение современных технологий защиты веб-приложений. Практическая часть включала глубокий анализ актуальных угроз информационной безопасности через призму методологий MITRE ATT&CK и OWASP, что позволило систематизировать знания о современных векторах атак и методах защиты. Особую ценность имел разбор реального инцидента кибербезопасности, который наглядно продемонстрировал тактики злоумышленников и важность своевременного выявления угроз.

Сравнительное описание современных методов защиты ПО позволило понять, как разработчики ПО могут до и после выпуска обнаруживать уязвимости и далее – устранять их.

Значимым для углубления в сферу информационной безопасности стало участие в профильных мероприятиях и мастер-классах. Полученные результаты демонстрируют, что сочетание технических исследований с просветительской работой способствует как повышению уровня цифровой грамотности, так и развитию практических навыков в области кибербезопасности.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. **MITRE ATT&CK®** [Электронный ресурс]. – Режим доступа: <https://attack.mitre.org/> (дата обращения: 21.04.2025).
2. **OWASP Foundation** [Электронный ресурс]. – Режим доступа: <https://owasp.org/> (дата обращения: 21.04.2025).
3. **GitHub Docs** [Электронный ресурс]. – Режим доступа: <https://docs.github.com/> (дата обращения: 21.04.2025).
4. **Markdown Guide** [Электронный ресурс]. – Режим доступа: <https://www.markdownguide.org/> (дата обращения: 21.04.2025).
5. **Исследование киберугроз 2024 года** / Аналитический отчёт Group-IB. – 2024. – 45 с.
6. **OWASP Top 10:2021** [Электронный ресурс]. – Режим доступа: <https://owasp.org/www-project-top-ten/> (дата обращения: 21.04.2025).