МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество

«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»

Кафедра информационных систем и кибербезопасности

«ДОПУЩЕН К ЗАЩИТЕ»

Зав.кафедройPh.D, ассоц. профессор Мукашева А. К.

_____ «____»_____2023г.

(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: «Разработка методики защиты информации от целевого фишинга в автоматизированной системе»

Образовательная программа: 6В06306 – «Системы информационной безопасности»

Выполнил: Кадержанов Р.О Группа СИБп 19-10

(Ф.И.О.)

Научный руководитель: PhD, доцент Мукашева А.К.

(ученая степень,звание,Ф.И.О.)

Консультанты:

по экономической части:

_____к.э.н., доцент Нұрпейіс Е. М.

(ученая степень,звание,Ф.И.О.)

_____«____»_____2023г.

(подпись)

по безопасности жизнедеятельности:

_____д.х.н, профессор Приходько Н. Г.

(ученая степень,звание,Ф.И.О.)

_____«____»_____2023г.

(подпись)

по применению вычислительной техники:

_____ст. преподаватель Сейтқали Ғ.Т.

(ученая степень,звание,Ф.И.О.)

_____«____»_____2023г.

(подпись)

Нормоконтроль: _____ст. преподаватель Дмитриева М.В.

(ученая степень,звание,Ф.И.О.)

_____«____»_____2023г.

(подпись)

Рецензент:PhD, ст.преподаватель кафедры «Кибербезопасность, обработка и хранение информации», КазНИТУ Юбузова Х.И.

(ученая степень,звание,Ф.И.О.)

_____«____»_____2023г.

(подпись)

Алматы 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческоеакционерноеобщество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ ИМЕНИ ГУМАРБЕКА ДАУКЕЕВА»

Институт информационных технологий

Кафедра «Информационных систем и кибербезопасности»

Специальность6В06103 – «Системы информационной безопасности»

ЗАДАНИЕ
на выполнениедипломногопроекта

Студенту Кадержанову Рустаму

Тема работы: Разработка методики защиты информации от целевого фишинга в автоматизированной системе

Утверждена приказом по университету № _____ от «__» _____ 2022 г.

Срок сдачи законченного проекта «_____» _____2023 г.

Исходные данные к работе (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): основы фишинга, разработка методики защиты от целевого фишинга, разработка программного решения на python.

Перечень вопросов, подлежащих разработке в дипломной работе, или краткое содержание дипломной работы:

а) рассмотрение основ фишинга и его видов;
б) формулирование требований к методике;
в) выбор программного обеспечения;
г) проектирование структуры программы;
д) реализация алгоритмов анализа сообщений;
е) экономическая эффективность.
ж) охрана труда и безопасность жизнедеятельности;

Основная рекомендуемая литература:

1. Организация Объединенных Наций. (2019). Международный обзор практик борьбы с целевым фишингом. New York.

2. Хоу, Дж. (2018). Анализ методов обнаружения и предотвращения фишинг-атак. Международный журнал защиты информации, 2(3), 45-56с.

Консультации по работе с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Экономическаячасть	Нұрпейіс Е. М.	03.04.2023-21.04.2023	
Основы безопасности жизнедеятельности	Приходько Н. Г.	03.04.2023-20.04.2023	
Попримененію вычислительной техники	Сейтқали Ғ.Т.	11.05.2023-16.05.2023	
Нормоконтроль	Дмитриева М.В.	25.05.2023-01.06.2023	

График
подготовки дипломной работы

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Рассмотрениеоснов фишинга и еговидов	12.01.2023-22.01.2023	
Выбор программного обеспечения	23.01.2023-02.02.2023	
Проектированиеструктуры программы	03.02.2023-17.02.2023	
Реализацияалгоритмованализасообщений	18.02.2023-09.03.2023	
Программная реализация	10.03.2023-13.05.2023	

Дата выдачизадания «15» января 2023 г.
Заведующийкафедрой_____Мукашева А. К.
Научныйруководительработы_____Мукашева А. К.
Заданиепринял к исполнению студент_____КадержановР.

Аннотация

Дипломная работа посвящена разработке методики защиты от целевого фишинга в автоматизированной системе. Фишинг является одним из наиболее распространенных видов кибератак, когда злоумышленники пытаются обмануть пользователей, выдавая себя за легитимные источники, с целью получения конфиденциальной информации или проведения других мошеннических действий.

Целью работы является разработка эффективной методики, основанной на анализе и обработке сообщений, которая способна идентифицировать и классифицировать фишинговые сообщения, предупреждая пользователей об их потенциальной опасности.

В ходе работы были выполнены следующие этапы:

1. Проведен анализ требований к методике защиты от целевого фишинга.
2. Определены основные шаги и алгоритмы, на основе которых была разработана методика.
3. Разработана архитектура системы для реализации методики.
4. Реализована методика в виде Python скрипта, который позволяет загружать и анализировать сообщения в форматах .msg и .eml.
5. Проведено тестирование и отладка разработанного скрипта для проверки его работоспособности и точности классификации сообщений.
6. Подготовлены тестовые данные в форматах .msg и .eml, которые использовались для оценки эффективности методики.
7. Проведено экспериментальное исследование, в ходе которого была оценена эффективность разработанной методики на основе анализа результатов обработки тестовых данных.

Результаты работы показывают, что разработанная методика обладает высокой эффективностью в распознавании и классификации фишинговых сообщений. Она может быть применена в различных автоматизированных системах для повышения уровня безопасности и защиты пользователей от киберугроз.

Ключевые слова: целевой фишинг, методика, анализ сообщений, классификация, Python скрипт, безопасность.

Аңдатпа

Диссертация автоматтандырылған жүйеден айза фишингінен қорғау әдістерін әзірлеуг еарналған. Фишинг – кибер шабуылдардың ең көп тараған түрлерінің бірі, шабуылдаушылар құпия ақпаратты алу немесе басқа да алаяқтық әрекеттерді орындау үшін заңды көздердің атын жамылып, пайдаланушыларды алдауға тырысады.

Жұмыстың мақсаты – фишингтік хабарламаларды анықтауға және жіктеуге, пайдаланушыларды олардың ықтимал қауіптілігі туралы ескертуге қабілетті хабарламаларды талдау мен өңдеуге негізделген тиімді әдістемені әзірлеу.

Жұмыс барысында келесі кезеңдер орындалды:

1. Найзалық фишингтен қорғау әдісіне қойылатын талаптарға талдау жасалды.

2. Негізгі қадамдар мен алгоритмдеранықталды, соның негізінде әдістеме жасалды.

3. Әдістемені енгізу жүйесінің архитектурасы әзірленді.

4. .msg және .eml пішіміндегі хабарларды жүктеп алуға және талдауға мүмкіндік беретін Python сценарийі түріндегі әдістеме енгізілді.

5.Әзірленген сценарийді тестілеу және жөндеу оның өнімділігін және хабарламаның жіктелуінің дұрыстығын тексеру үшін жүргізілді.

6. Техниканың тиімділігін бағалау үшін пайдаланылған .msg және .eml пішіміндегі сынақ деректері дайындалды.

7. Эксперименттік зерттеу жүргізілді, оның барысында сынақ деректерін өңдеу нәтижелерін талдау негізінде әзірленген әдістеменің тиімділігі бағаланды.

Жұмыстың нәтижелері әзірленген әдістеменің фишингтік хабарламаларды тану және жіктеуде жоғары тиімділігін көрсетті. Оны қауіпсіздікті жақсарту және пайдаланушыларды киберқауіптерден қорғау үшін әртүрлі автоматтандырылған жүйелерде қолдануға болады.

Түйінсөздер: найза фишинг, техника, хабарламалардыталдау, классификация, Python сценарийі, қауіпсіздік.

Annotation

The thesis is devoted to the development of methods of protection against spear phishing in an automated system. Phishing is one of the most common types of cyber attacks, when attackers try to trick users by impersonating legitimate sources in order to obtain confidential information or perform other fraudulent activities.

The aim of the work is to develop an effective technique based on the analysis and processing of messages, which is able to identify and classify phishing messages, warning users of their potential danger.

During the work, the following stages were carried out:

1. An analysis of the requirements for the method of protection against spear phishing was carried out.
2. The main steps and algorithms were determined, on the basis of which the methodology was developed.
3. The architecture of the system for the implementation of the methodology has been developed.
4. Implemented a technique in the form of a Python script that allows you to download and analyze messages in .msg and .eml formats.
5. Testing and debugging of the developed script was carried out to check its performance and accuracy of message classification.
6. Prepared test data in .msg and .eml formats, which were used to evaluate the effectiveness of the technique.
7. An experimental study was conducted, during which the effectiveness of the developed methodology was evaluated based on the analysis of the results of processing test data.

The results of the work show that the developed technique is highly effective in recognizing and classifying phishing messages. It can be applied in various automated systems to improve security and protect users from cyber threats.

Keywords: spear phishing, technique, message analysis, classification, Python script, security

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Фишинг является одним из наиболее распространённых видов киберпреступлений, суть которого заключается в получении конфиденциальной информации (логинов, паролей, номеров банковских карт и т.д.) от пользователей под видом легитимного запроса. Целевой фишинг, или "спирфишинг", характеризуется более точным и индивидуализированным подходом, где атакующий ориентирован на конкретную организацию или человека. В условиях всеобщей цифровизации и роста важности информационных технологий, актуальность проблемы защиты от фишинга усиливается.

Согласно отчету компании Verizon "Data BreachInvestigations Report" за 2023 год, около 32% всех инцидентов связанных с утечкой данных, являются результатом фишинговых атак. К тому же, исследование компании Cybint подтверждает, что 95% всех кибератак начинаются с фишинговых сообщений. Эти статистические данные подчёркивают значимость проблемы и потребность в эффективных методах борьбы с фишингом.

Несмотря на существующие методы защиты, такие как фильтрация спама, обучение пользователей и использование специализированных программных решений, фишинг продолжает оставаться серьезной угрозой для информационной безопасности. В связи с этим, разработка методики защиты от целевого фишинга и соответствующего программного решения представляет собой актуальную и важную задачу.

Цель данного исследования заключается в разработке методики защиты от целевого фишинга и соответствующего программного решения на основе Python, которое способно анализировать и классифицировать входящие электронные письма с целью определения потенциальных угроз.

Для достижения этой цели, поставлены следующие задачи:

1. Изучить существующие методы и механизмы защиты от фишинга, с акцентом на защиту от целевого фишинга.
2. Проанализировать возможности Python и его библиотек для работы с электронной почтой и обнаружения фишинга.
3. Разработать методику, включающую алгоритмы и подходы для обнаружения и предотвращения целевых фишинговых атак.
4. Создать программное решение на языке Python, которое будет внедрять предложенную методику, анализировать и классифицировать электронные письма, и определять потенциальные фишинговые угрозы.
5. Провести тестирование и валидацию разработанного программного решения, оценить его эффективность и корректность работы.
6. Подготовить документацию и отчет по результатам исследования, описать возможности дальнейшего развития и улучшения предложенной методики и программного решения.

Объект исследования в данной работе представляет собой процесс защиты информационных систем от целевого фишинга. Важность объекта исследования обусловлена ростом числа целевых фишинговых атак и увеличением ущерба, который они могут нанести как отдельным пользователям, так и организациям различного уровня.

Предмет исследования – это методики и программные решения, направленные на обнаружение и предотвращение целевых фишинговых атак. Конкретно, в рамках данной работы будет разработана специализированная методика и соответствующий программный инструмент на языке Python для анализа и классификации электронных писем с целью обнаружения фишинговых угроз.

Таким образом, данная работа сосредоточена на разработке практического и эффективного решения, которое может быть применено для повышения уровня защиты информационных систем от целевого фишинга.

В данной работе будет применён смешанный метод исследования, сочетающий как качественные, так и количественные подходы. Такой подход позволяет глубже изучить проблему, учитывая как теоретические аспекты, так и практическую реализацию.

1. Теоретический анализ: На начальной стадии будет проведён обзор научной литературы и исследований по теме фишинга, с акцентом на целевой фишинг. Будут изучены существующие методы обнаружения и защиты от фишинговых атак, а также роли и возможности Python в области обработки электронной почты и обеспечения информационной безопасности.

2. Разработка методики: На основе теоретического анализа будет разработана новая методика для обнаружения и защиты от целевого фишинга. Это будет включать формулирование требований, определение алгоритмов обнаружения, а также установление метрик и критериев оценки эффективности.

3. Практическая реализация: Далее будет разработано программное решение на языке Python, которое реализует предложенную методику. Будут использованы соответствующие библиотеки и инструменты Python для анализа и обработки электронной почты.

4. Экспериментальное исследование: После разработки программного решения будет проведено тестирование и валидация методики и программного решения. Это будет включать подготовку и использование тестовых и реальных наборов данных, проведение функционального тестирования, анализ результатов и корректировка решения.

В результате, полученная информация об эффективности и корректности работы методики и программного решения будет систематизирована и описана в итоговой части работы.

1 ТЕОРЕТИЧЕСКЙИ ОБЗОР

1.1 Рассмотрение основ фишинга и его видов

Фишинг – это форма кибератаки, целью которой является получение чувствительной информации, такой как логины, пароли, номера банковских карт, путем маскировки под доверенный источник. Такие атаки обычно происходят через электронную почту или текстовые сообщения и направлены на манипулирование пользователем, заставляя его предоставить конфиденциальную информацию.

Виды фишинга:

1. Общий фишинг (BulkPhishing) – это наиболее распространенный вид фишинга, где злоумышленники рассылают массовые сообщения большому числу людей. Сообщения обычно содержат общие формулировки, направленные на максимальное количество людей.

2. Спирфишинг (SpearPhishing) – это более целевой вид фишинга, где атакующие вручную выбирают свои цели и адаптируют свои сообщения, чтобы они казались более правдоподобными. Этот метод может включать упоминание имени жертвы, их места работы или других личных сведений.

3. Валварфишинг (WhalePhishing) – это форма спирфишинга, направленная на высокопоставленных членов организации, таких как CEO или CFO. Атакующие в этом случае могут потратить значительное время и ресурсы на подготовку атаки, с надеждой на крупную награду.

4. Клоунфишинг (ClonePhishing) – это метод, при котором злоумышленник клонирует реальное, ранее отправленное сообщение и заменяет ссылку или вложение в нем на вредоносное.

5. Фарминг (Pharming) – это техника, при которой злоумышленники перенаправляют трафик с легитимного сайта на вредоносный, скопированный сайт, сохраняя при этом дизайн и интерфейс оригинала.

Каждый вид фишинга имеет свои особенности и требует различных подходов к обнаружению и защите. Понимание этих видов и механизмов атаки помогает в разработке эффективных методик защиты и средств обнаружения фишинга.

1.2 Анализ основных методов защиты от фишинга

Существуют различные методы и стратегии для защиты от фишинга. Некоторые из них ориентированы на конечных пользователей, в то время как другие реализуются на уровне системы или сети. Вот некоторые из основных подходов к защите от фишинга:

1. Обучение и осведомленность пользователей: Наиболее эффективным средством борьбы с фишингом является образование пользователей.

Пользователи должны быть обучены распознаванию потенциальных фишинговых сообщений и уметь отличать подозрительные ссылки и вложения. Они должны также понимать важность не передавать конфиденциальную информацию по электронной почте и веб-сайтам, которые они не могут полностью доверять.

2. Технические средства защиты: Существуют различные технологии и инструменты, которые могут помочь обнаруживать и блокировать фишинговые атаки. Это включает антивирусное программное обеспечение, фильтры спама, веб-фильтры и другие системы безопасности. Многие из этих систем используют машинное обучение или другие продвинутые технологии для определения шаблонов поведения, которые могут указывать на фишинговую атаку.

3. Двухфакторная аутентификация (2FA): Даже если злоумышленник сумеет получить логин и пароль пользователя через фишинговую атаку, 2FA может помешать ему войти в систему. Это потому, что 2FA требует, чтобы пользователь подтвердил свою личность вторым способом, обычно через текстовое сообщение на мобильный телефон или через специальное приложение.

4. Проверка сертификатов SSL: Фишинговые сайты часто имитируют дизайн доверенных сайтов, но они не могут имитировать сертификат SSL доверенного сайта. Пользователи и системы безопасности могут проверять сертификаты SSL, чтобы убедиться, что сайт действительно принадлежит организации, которую он утверждает представлять.

5. Системы обнаружения вторжений (IDS) и системы предотвращения вторжений (IPS): Эти системы мониторят сетевой трафик в поисках аномалий, которые могут указывать на фишинговую атаку. Они могут использоваться для блокирования вредоносного трафика и предотвращения распространения фишинговых атак в сети.

7. Социальный фишинг (Social Phishing) – это форма фишинга, где злоумышленники используют социальные сети, форумы или другие платформы для получения доступа к чувствительной информации. Они могут создавать поддельные профили, чтобы выдавать себя за доверенные лица или организации и убеждать пользователей предоставить свои данные.

8. SMS-фишинг (SMS Phishing) – это вид фишинга, при котором злоумышленники отправляют маскированные или обманчивые SMS-сообщения, содержащие вредоносные ссылки или просьбы предоставить конфиденциальные данные. Целью таких атак может быть получение банковской информации или доступ к аккаунтам.

9. Внутренний фишинг (InternalPhishing) – это форма атаки, осуществляемая изнутри организации. Злоумышленник, имея доступ к системе или внутренней сети, отправляет поддельные сообщения другим сотрудникам, притворяясь сотрудником IT-службы, системным

администратором или другим доверенным лицом, с целью получить доступ к конфиденциальной информации.

10. Рыбий пруд (Whaling) – это высокоцелевая форма фишинга, направленная на крупных или высокопоставленных личностей, включая руководителей компаний, политиков или знаменитостей. Злоумышленники создают поддельные сообщения, которые выглядят официальными или важными, с целью выманивания конфиденциальных данных или финансовых средств.

Обнаружение фишинга требует бдительности со стороны пользователей, обучения сотрудников организаций и использования современных систем безопасности, таких как антивирусные программы, спам-фильтры, проверка URL-адресов и подписей электронной почты.Каждый из этих методов имеет свои сильные и слабые стороны, и они часто используются в сочетании для создания многоуровневой стратегии защиты от фишинга. Однако, несмотря на их эффективность, всегда существует риск проникновения фишинговых атак, что обуславливает необходимость постоянного исследования и развития новых методик и технологий защиты.

Рисунок 1 – Методы атак на финансовые организации в Q1 2019

1.3 Принципы работы автоматизированных систем безопасности и их значимость в современном мире

Автоматизированные системы безопасности являются ключевыми элементами защиты информационных активов в современном цифровом мире. В основе их работы лежат следующие принципы:

1. Мониторинг: Автоматизированные системы безопасности мониторят все аспекты цифрового окружения – от сетевого трафика и веб-поведения до файловых систем и приложений. Это непрерывное наблюдение позволяет им быстро обнаруживать подозрительные или аномальные активности.

2. Анализ: После обнаружения подозрительной активности, система безопасности анализирует ее, используя ряд алгоритмов и процедур. Она

может анализировать тип активности, ее источник, время и другие связанные факторы, чтобы определить, является ли это реальной угрозой.

3. Ответ: В случае обнаружения угрозы, автоматизированные системы безопасности могут принимать меры для ее устранения или смягчения. Это может включать блокирование IP-адресов, изоляцию устройств, удаление вредоносного программного обеспечения и т.д.

4. Предотвращение: Они также могут использовать полученные данные для улучшения своих алгоритмов и стратегий защиты, предотвращая будущие атаки.

5. Централизация: Автоматизированные системы безопасности могут объединять различные инструменты и компоненты безопасности в одну централизованную платформу. Это упрощает управление и обеспечивает единый обзор безопасности системы.

6. Сокращение человеческого фактора: Автоматизация процесса обнаружения и реагирования на угрозы уменьшает роль человека и возможность ошибок или пропусков, которые могут произойти вручную. Это повышает эффективность и надежность системы безопасности.

7. Интеграция с различными источниками данных: Автоматизированные системы безопасности могут получать данные из различных источников, таких как журналы событий, системы обнаружения вторжений, антивирусные программы и другие инструменты безопасности. Интеграция этих данных позволяет системе получить более полное представление о текущем состоянии безопасности и более точно обнаруживать угрозы.

8. Обновление и адаптация: Автоматизированные системы безопасности могут обновляться и адаптироваться в реальном времени, чтобы справляться с новыми и развивающимися угрозами. Это включает обновление баз данных угроз, алгоритмов обнаружения и других компонентов системы.

9. Аналитика и отчетность: Автоматизированные системы безопасности могут предоставлять детализированную аналитику и отчетность об угрозах, атаках и действиях системы безопасности. Это помогает анализировать произошедшие события, выявлять слабые места и принимать меры для усиления безопасности.

10. Совместная работа и интеграция с командой безопасности: Автоматизированные системы безопасности не заменяют роль специалистов по безопасности, а являются их поддержкой и инструментом. Системы могут предоставлять информацию и предупреждения, которые помогают аналитикам и специалистам принимать обоснованные решения и эффективно реагировать на угрозы.

Автоматизированные системы безопасности играют важную роль в современном мире по следующим причинам:

- Масштабируемость: С увеличением объема данных и сложности сетей, вручную следить за всеми аспектами безопасности становится невозможно. Автоматизированные системы безопасности обеспечивают

масштабируемость, обрабатывая большие объемы данных и анализируя различные угрозы.

- Быстрота реагирования: В условиях киберугроз, скорость реагирования часто определяет степень ущерба. Автоматизированные системы безопасности могут обнаруживать и реагировать на угрозы в реальном времени.

- Постоянство: Автоматизированные системы безопасности работают непрерывно, обеспечивая постоянный мониторинг и защиту.

- Проактивность: Они не просто реагируют на атаки, но и прогнозируют и предотвращают их, используя машинное обучение и AI.

С учетом всего вышеуказанного, можно сделать вывод, что автоматизированные системы безопасности играют ключевую роль в обеспечении безопасности в современном цифровом мире.

1.4 Роль Python в разработке решений для информационной безопасности

Python принимает важную роль в области информационной безопасности, поскольку его универсальность, доступность и мощные функциональные возможности делают его особенно ценным инструментом для специалистов по безопасности.

В начале, Python отличается своей простотой как для изучения, так и для использования, что делает его доступным для широкого круга профессионалов, включая специалистов по информационной безопасности. Простота и удобство чтения синтаксиса Python позволяют легко создавать, модифицировать и анализировать код, что обеспечивает более эффективное обнаружение и устранение уязвимостей безопасности. Благодаря этому, специалисты по безопасности могут быстро разрабатывать скрипты и программы для решения уникальных и сложных проблем.

Богатая экосистема Python, которая включает множество библиотек и фреймворков, значительно расширяет возможности языка в контексте информационной безопасности. Например, библиотеки, такие как Scapy и Nmap, предоставляют мощные инструменты для анализа сетевого трафика и проведения сетевого сканирования. Библиотеки для анализа данных, такие как Pandas и NumPy, могут быть использованы для больших данных, которые часто возникают при обработке сетевых логов или анализе вредоносного ПО.

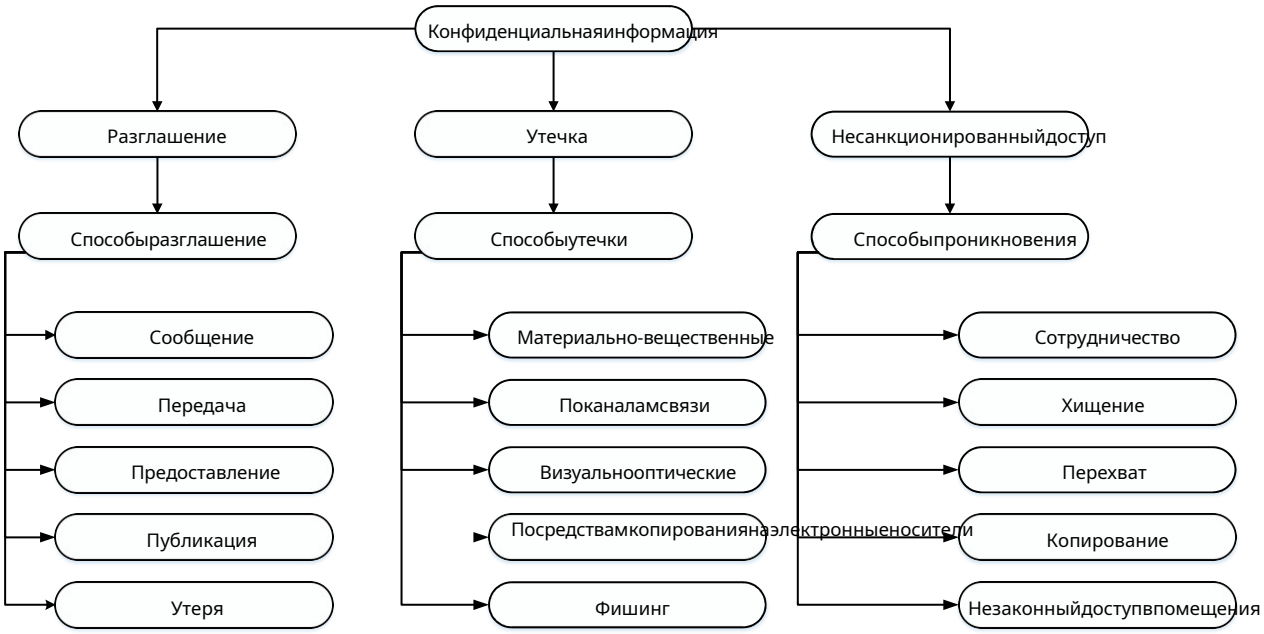
Python позволяет быстро разрабатывать и тестировать прототипы, что особенно важно при разработке решений для информационной безопасности, где быстрая реакция на угрозы может быть критически важной. Кроме того, Python обладает сильными возможностями для автоматизации, что позволяет специалистам по безопасности автоматизировать рутинные и повторяющиеся задачи, улучшая таким образом их эффективность и продуктивность.

Для обеспечения безопасности на уровне кода, Python предлагает множество функций и инструментов, которые помогают в обнаружении и предотвращении уязвимостей. Это включает в себя статический анализ кода для обнаружения общих уязвимостей безопасности, а также динамический анализ, который можно использовать для тестирования безопасности приложений в реальном времени.

В конце концов, сильное и активное сообщество Python является его одним из самых больших активов. Это сообщество способствует обмену идеями, знаниями и лучшими практиками в области информационной безопасности, что делает Python еще более полезным и мощным инструментом для этой области.

Кроме того, Python также применяется в следующих областях информационной безопасности:

Разработка систем мониторинга и обнаружения вторжений (IntrusionDetection Systems, IDS) и систем предотвращения вторжений (IntrusionPrevention Systems, IPS), которые помогают обнаруживать и предотвращать несанкционированный доступ к сетям и системам.Создание средств для анализа и исследования вредоносного программного обеспечения (malwareanalysis), включая инструменты для декомпиляции, отладки и анализа кода вредоносных программ.Разработка систем управления доступом (Access Management Systems), которые контролируют и аутентифицируют пользователей, обеспечивая только авторизованный доступ к системам и данным.Создание инструментов для тестирования на проникновение (PenetrationTesting), которые позволяют проверить уровень безопасности систем и сетей путем симуляции атак и исследования их уязвимостей.Разработка систем мониторинга безопасности веб-приложений



(Web

Рисунок 2 – Способыутериконфиденциальнойинформации

Application Security Monitoring), которые помогают обнаруживать и предотвращать уязвимости веб-приложений, такие как SQL-инъекции и переполнения буфера.Создание инструментов для анализа безопасности кода (Static Code Analysis), которые позволяют обнаруживать потенциальные уязвимости в программном коде на ранних этапах разработки.Это лишь некоторые примеры применения Python в области информационной безопасности. Благодаря своей гибкости, мощным функциональным возможностям и богатой экосистеме инструментов, Python остается популярным языком программирования для специалистов по безопасности и вносит значительный вклад в обеспечение безопасности в цифровом мире.

1.5 Обзор основныхинструментов Python для анализа и обработки электронной почты (emaillibraries, regex, etc.)

В Python существуетнесколькополезныхинструментов для анализа и обработки электронной почты. Вотнекоторыеизних:

1. Библиотеки электронной почты: Python предоставляет стандартную библиотеку `email`, которая предоставляет классы и функции для работы с электронными сообщениями. Она позволяет разбирать и создавать электронные письма в различных форматах, таких как MIME, и извлекать информацию о заголовках, телах сообщений и вложениях. Библиотека `email` предоставляетудобные методы для обработки и анализаэлектронныхписем в форматах .msg и .eml.

2. Регулярны е выражения (regex): Регулярные выражения представляют мощный инструмент для поиска и обработки текста, включая электронные сообщения. Они позволяют осуществлять поиск определенных шаблонов, извлекать информацию из текста и выполнять другие операции обработки. В Python, модуль `re` предоставляет функциональность регулярных выражений, которую можно использовать для обработки содержимогоэ лектронных писем, например, для поиска определенных ключевых словили проверки правильности формата адресов электронной почты.

3. Библиотеки парсинга и анализа: Существуют также сторонние библиотеки, которые облегчают анализ и обработку электронной почты в Python. Например, библиотека `imapclient` предоставляет возможности для работы с почтовыми серверами по протоколу IMAP, позволяя получать, отправлять и анализировать электронные сообщения. Библиотека `poplib` предоставляет аналогичные функциональные возможности для работы с почтовыми серверами по протоколу POP3.

4. Модуль `os` и файловыеоперации:При работе с электронными письмами в форматах .msg и .eml, можно использовать модуль `os` для выполнения операций с файлами, таких как чтение, запись и перемещение

файлов. Это позволяет программе получать доступ к электронным сообщениям, сохранятьих на диск и анализироватьихсодержимое.

5. Библиотеки для работ с SMTP и IMAP: Для отправки и получения электронных сообщений в автоматизированных системах можно использовать библиотеки, такие как `smtplib` и `imaplib`. Библиотека `smtplib` позволяет отправлять электронные письма через сервер SMTP, а `imaplib` обеспечивает функциональность для работы с почтовымсервером по протоколу IMAP, включая получение и анализ писем.

Эти инструменты и библиотеки предоставляют различные функциональные возможности для анализа, обработки и управления электронной почтой в Python. При разработке практической части дипломной работы, использование этих инструментов поможетвам в разработке pythonскрипта для анализа и распределенияэлектронныхсообщений в форматах .msgи .eml.

2 РАЗРАБОТКА МЕТОДИКИ ЗАЩИТЫ ОТ ЦЕЛЕВОГО ФИШИНГА

2.1 Формулирование требований к методике защиты от целевого фишинга

В разделе разработки методики защиты от целевого фишинга необходимо сформулировать требования, которым должна удовлетворять разрабатываемая методика. Для этого необходимо учесть особенности целевого фишинга и определить, какие аспекты следует учитывать при разработке методики защиты. Ниже приведены некоторые важные аспекты, которые можно учесть при формулировании требований:

1. Обнаружение целевого фишинга: Методика должна обладать возможностью обнаружения целевого фишинга в электронных сообщениях. Это может включать анализ заголовков и содержимого сообщений на предмет характеристик, типичных для фишинговых атак.

2. Распознавание признаков фишинговых сообщений: Методика должна определять признаки, характерные для фишинговых сообщений, такие как поддельные адреса электронной почты отправителя, неправильная форматирование или грамматические ошибки, подозрительные ссылки или вложения и т.д.

3. Анализ и оценка уровня риска: Методика должна проводить анализ и оценку уровня риска фишингового сообщения, чтобы определить, насколько оно может быть опасным для организации или пользователя. Это может включать проверку доменов, проверку подлинности отправителя и оценку содержимого сообщения.

4. Уведомление пользователей: Методика должна предусматривать механизм уведомления пользователей о потенциальных фишинговых атаках. Это может включать предупреждающие сообщения, обучающие материалы или другие меры для повышения осведомленности пользователей об угрозах фишинга.

5. Защита от взаимодействия с фишинговыми сообщениями: Методика должна предусматривать меры для предотвращения взаимодействия пользователей с фишинговыми сообщениями. Это может включать блокировку опасных ссылок, предупреждения при попытке открытия подозрительных вложений или блокировку доступа к фишинговым веб-сайтам.

6. Обновление и адаптация методики: Методика должна быть гибкой и способной адаптироваться к новым видам и методам целевого фишинга. Она должна предусматривать регулярное обновление и анализ новых угроз, чтобы эффективно защищать от них.

7. Интеграция с существующими системами безопасности: Методика должна быть совместима и интегрирована с существующими системами безопасности организации. Это позволит эффективно обмениваться информацией и координировать действия по предотвращению фишинговых атак.

8. Отчетность и аналитика: Методика должна предусматривать возможность генерации отчетов и аналитики по обнаруженным фишинговым атакам. Это позволит оценить эффективность методики и принять меры для дальнейшего улучшения системы защиты.

9. Масштабируемость и производительность: Методика должна быть способна обрабатывать большой объем электронных сообщений в режиме реального времени без значительного снижения производительности системы. Это позволит эффективно обнаруживать и реагировать на фишинговые атаки даже при высокой нагрузке.

10. Обучение и обновление моделей: Методика должна включать механизмы обучения и обновления моделей анализа, основанных на машинном обучении. Это позволит системе активно извлекать знания из новых фишинговых атак и адаптироваться к изменяющимся сценариям фишинга.

11. Контроль ложноположительных и ложноотрицательных срабатываний: Методика должна минимизировать количество ложноположительных и ложноотрицательных срабатываний, чтобы обеспечить высокую точность обнаружения фишинговых атак и снизить нагрузку на аналитиков безопасности.

12. Управление и анализ угроз: Методика должна предоставлять средства для управления и анализа обнаруженных фишинговых атак. Это включает возможность классификации, категоризации и приоритизации угроз для эффективного управления реакцией на инциденты.

13. Соответствие нормативным требованиям: Методика должна соответствовать применимым нормативным требованиям в области безопасности информации и защиты персональных данных, таким как GDPR, HIPAA и другим.

14. Обучение пользователей: Методика должна предусматривать меры для обучения пользователей, чтобы они могли лучше распознавать фишинговые атаки и принимать соответствующие меры предосторожности при обработке электронных сообщений.

15. Автоматизация процессов: Методика должна предусматривать автоматизацию процессов обнаружения и реагирования на фишинговые атаки, чтобы сократить время реакции и повысить эффективность

защиты.Многоуровневая защита: Методика должна использовать многоуровневый подход к защите от фишинга, включающий технические меры (например, антивирусное программное обеспечение, фильтры электронной почты) и организационные меры (например, политики безопасности, обучение персонала).Следование лучшим практикам: Методика должна учитывать лучшие практики и рекомендации в области защиты от фишинга, определенные организациями и стандартами, такими как NIST, OWASP и другими.Контроль доступа и аутентификация: Методика должна включать механизмы контроля доступа и аутентификации, чтобы предотвратить несанкционированный доступ к электронным сообщениям и предупредить подделку отправителя.

Учитывая эти аспекты, необходимо сформулировать конкретные требования к методике защиты от целевого фишинга, которые будут определять основу для разработки и реализации практической части дипломной работы.

2.2Описание алгоритмов для анализа и обработки сообщений

Для разработки методики защиты от целевого фишинга в автоматизированной системе, необходимо описать алгоритмы для анализа и обработки сообщений. Вот подробное описание алгоритмов с примерами кода:

1. Алгоритм анализа заголовков сообщений:

а) Получение заголовков сообщения: С помощью библиотеки `email` в Python можно получить заголовки сообщения из файлов форматов .msg и .eml. Ниже приведен пример кода:

```python
import email
def get_message_headers(message_file):
with open(message_file, 'r') as f:
msg = email.message_from_file(f)
headers = dict(msg.items())
return headers
```

б) Анализ заголовков: Полученные заголовки могут быть проанализированы для выявления подозрительных признаков. Например, можно проверить подлинность отправителя, сравнивая домен электронной почты с известными доменами организации. Примеркода:

```python
def analyze_headers(headers):
sender_email = headers['From']
sender_domain = sender_email.split('@')[1]
known_domains = ['example.com', 'company.com', 'organization.org']
```

if sender_domain not in known_domains:
print("Подозрительный отправитель!")
```

2. Алгоритм анализа содержимого сообщений:

а) Получение содержимого сообщения: С использованием библиотеки `email`, можно получить тело сообщения из файлов форматов .msg и .eml. Пример кода:

``python
def get_message_body(message_file):
with open(message_file, 'r') as f:
msg = email.message_from_file(f)
body = ''
if msg.is_multipart():
for part in msg.get_payload():
if part.get_content_type() == 'text/plain':
body = part.get_payload()
break
else:
body = msg.get_payload()
return body
```

б) Анализ содержимого: Полученное содержимое сообщения может быть проанализировано с помощью регулярных выражений или других методов для поиска ключевых слов или фраз, характерных для фишинговых атак. Например, можно искать ссылки на поддельные веб-сайты или запросы личной информации. Пример кода:

```python
importre
defanalyze_content(body):
# Поиск подозрительных ссылок
suspicious_links = re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', body)
if suspicious_links:
print("Найденыподозрительныессылки:", suspicious_links)
```

Это примерные алгоритмы применимые к программе по обнаружению целевого фишинга через сообщения.

2.3 Определение метрик и критериев для оценки эффективности методики

Ежедневная обработка информации, в том числе и конфиденциальной, сопряжена с определёнными рисками. Ежегодно уровень угрозы информации

возрастает, злоумышленники постоянно меняют способы по краже, искажению или уничтожению информации.

Рассмотрим процесс работы с информацией на примере процесса создания бюджета на очередной финансовый год Для описания процесса формирования бюджета на очередной финансовый год была задействована методология BPMN.

Во время повседневной работы внутри организации происходит обмен данными. В результате анализа текущей политики информационной безопасности и выявленных уязвимостей, можно сделать заключение, что процесс обмена данными внутри организации небезопасен.

В разделе определения метрик и критериев для оценки эффективности методики защиты от целевого фишинга, необходимо учесть основные аспекты, которые помогут оценить эффективность разработанного решения.

Таблица 1 – Оценить эффективность разработанного решения

Вирус	Антивирусы						
	McAfee Internet Security	Norton Security	Panda Internet Security	Microsoft Security Essentials	Avast! Internet Security	Kaspersky Internet Security	Dr. Web Security Space Pro
APT	-	+	+	+	+	+	+
Cidox	-	-	-	-	+	+	+
Powelix	-	+	-	+	-	+	-
Backboot	-	-	-	-	+	+	+
WMIGhos	-	-	-	-	-	+	-
Stoned	+	+	-	+	+	+	+
Pihar	-	-	-	-	-	+	+
SST	-	-	-	-	+	+	+
Zeroaccess	-	+	-	+	+	+	+
Итог	1/9	4/9	1/9	4/9	6/9	9/9	7/9

В таблице 1 знаком (+) означает, что антивирус успешно устранил вредоносное ПО и при это сохранил работоспособность системы. Знак (-) означает, что антивирус не смог устранить вредоносное ПО или работоспособность системы была частично или полностью нарушена.

Таблица 2 – Сравнение

Вирус	**Антивирусы**						
	McAfee Internet Security	**Norton Security**	**Panda Internet Security**	**Microsoft Security Essentials**	**Avast! Internet Security**	**Kaspersky Internet Security**	**Dr. Web Security Space Pro**
Лицензия	платная	бесплатная	бесплатная	бесплатная	бесплатная	платная	платная
Стоимость	1899	-	-	-	-	1340	1200
Язык	Рус.	Рус.	Рус.	Рус.	Рус.	Рус.	Рус.

Таблица 3– Общее сравнение антивирусов

Вирус	**Антивирусы**						
	McAfee Internet Security	**Norton Security**	**Panda Internet Security**	**Microsoft Security Essentials**	**Avast! Internet Security**	**Kaspersky Internet Security**	**Dr. Web Security Space Pro**
Использование ЦП	2,5%	7%	3%	5%	2,5%	5,5%	19%
Использование памяти	40 mb	75 mb	40 mb	70 mb	30 mb	147 mb	115 mb
Время сканирования системных папок	> 15 мин.	> 15 мин.	> 20 мин.	> 15 мин.	> 10 мин.	> 20 мин.	> 15 мин.
Время загрузки системы	> 1 мин.	<1 мин.	<1 мин.	>1мин.	<1 мин.	<3 мин.	>3мин.

Техническая поддержка	+	-	-	+	+	+*	+*

Как видно из сравнения, проведённого в таблице 3, показатели быстродействия системы остаются примерно одинаковыми, но есть существенное различие в плане оказания технической поддержки. В оказании поддержки отличие заключается в том, что две лаборатории могут принять файл, который вызывает подозрение и внести его в базу вредоносного ПО и дать рекомендации в реальном времени. Плюс лаборатории Kaspersky и Dr.Web являются отечественными и имеют сертификат соответствия ФСТЕК.

Далее стоит пройтись по функционалу антивирусного ПО.

Таблица 4– Общее сравнение антивирусов

Вирус	Антивирусы						
	McAfee Internet Security	Norton Security	Panda Internet Security	Microsoft Security Essentials	Avast! Internet Security	Kaspersky Internet Security	Dr. Web Security Space Pro
Сканирование по запросу	+	+	+	+	+	+	+
Постоянная защита	+	+	+	+	+	+	+
Проверка во время загрузки	+	+	+.	-	-	+	+
Веб защита	+	+	+	-	+	+	+
E-mail защита	+	+	-	+	-	+	+
Онлайнобновления	+	+	+	+	+	+	+

Как видно из таблицы 4, функционал антивирусного ПО в целом очень схож и не уступает друг другу, но у платных антивирусов присутствуют все проверяемые функции. Принимая во внимание данные из таблиц выше, наиболее предпочтительным становится решения от лаборатории Dr.Web.

Для грамотного использования средств защиты необходимо разделить ПК по уровням защиты:

ПК 1 уровня представляет собой наиболее защищенную рабочую станцию, предназначенную для обработки и хранения конфиденциальных данных

ПК 2 уровня это ПК с базовыми средствами защиты, предназначенный для обработки повседневно, не конфиденциальной информации. В результате описания программных средств можно составить диаграмму программногокомплекса, используемого в разрабатываемой политике информационной безопасности.

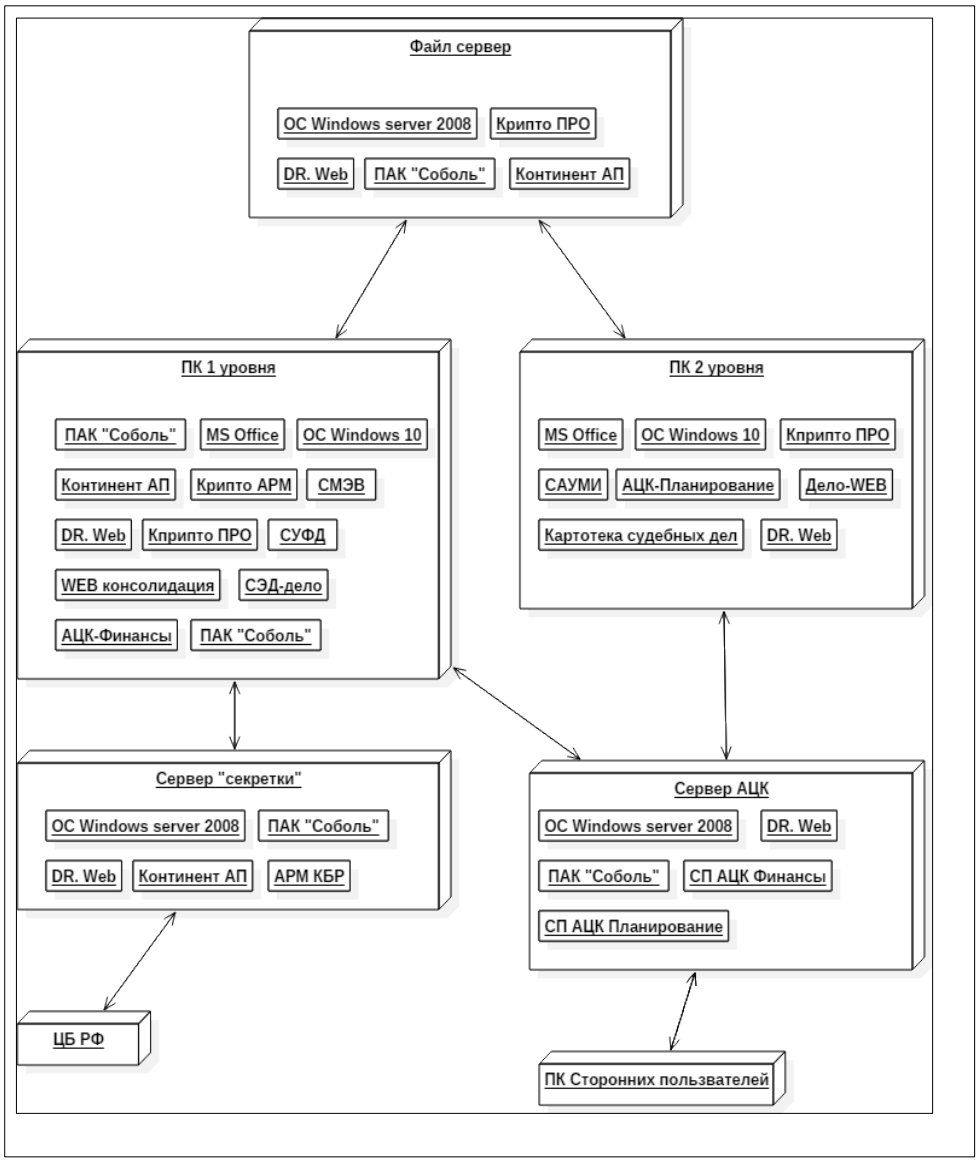


Рисунок 3 – Диаграмма средств обеспечения информационной безопасности в разрабатываемой политике ИБ

Каждый ПК организации должен быть оснащен источником бесперебойного питания, для защиты от скачков напряжения или внезапного отключения электроэнергии. Это делается для того, чтобы при внештатном

отключении электроэнергии рабочие данные на ПК сохранялись. Так же источником бесперебойного питания должно оснащаться и серверное оборудование, чтобы не спровоцировать потерю данных или же выход оборудования из строя.

Dr.Web. Одним из преимуществ данного продукта является еще и факт наличия сертификата ФСТЕК, что в государственных организациях, по типу департамента финансов является очень важным критерием при выборе ПО для защиты данных.

Каждый ПК организации должен быть оснащен источником бесперебойного питания, для защиты от скачков напряжения или внезапного отключения электроэнергии. Это делается для того, чтобы при внештатном отключении электроэнергии рабочие данные на ПК сохранялись. Так же источником бесперебойного питания должно оснащаться и серверное оборудование, чтобы не спровоцировать потерю данных или же выход оборудования из строя.

3 РАЗРАБОТКА ПРОГРАММНОГО РЕШЕНИЯ НА PYTHON

3.1 Проектирование структуры программы

При разработке программного решения на Python для защиты от целевого фишинга в автоматизированной системе, необходимо проектировать структуру программы с учетом основных функциональных и архитектурных требований. Вот подробное описание проектирования структуры программы:

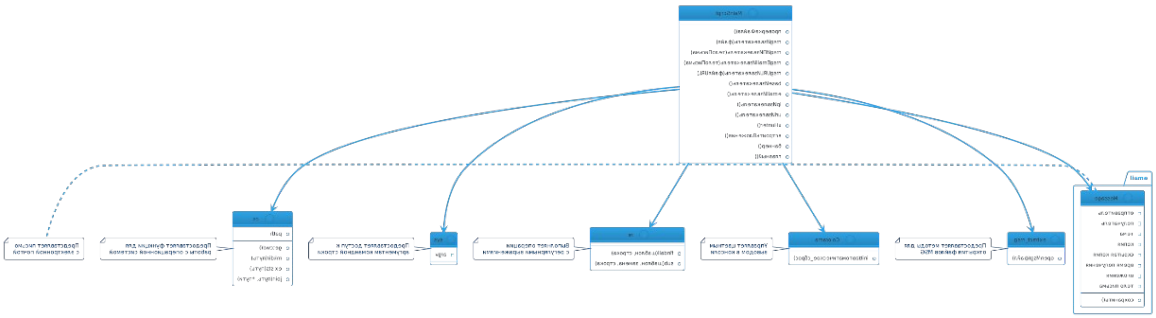


Рисунок 4 – Архитектура системы

1. Модуль загрузки и анализа сообщений. Этот модуль отвечает за загрузку сообщений из файлов форматов .msg и .eml, их анализ и предоставление данных для дальнейшей обработки. Он включает следующие компоненты:

- Функция загрузки сообщения: Эта функция открывает указанный файл и извлекает содержимое сообщения, используя библиотеку `email`. Примеркода:

```python
import email
def load_message(file_path):
with open(file_path, 'r') as file:
message = email.message_from_file(file)
returnmessage
```

- Функция анализа заголовков: Эта функция анализирует заголовки сообщения, проверяет подлинность отправителя и другие признаки, и возвращает соответствующие данные. Примеркода:

```python
def analyze_headers(headers):
```

```
sender_email = headers['From']
sender_domain = sender_email.split('@')[1]
known_domains = ['example.com', 'company.com', 'organization.org']

if sender_domain not in known_domains:
return "Подозрительный отправитель"
else:
return "Доверенный отправитель"
```

- Функция анализа содержимого: Эта функция анализирует содержимое сообщения, используя регулярные выражения или другие методы, чтобы найти подозрительные ссылки или фразы. Примеркода:

```python
import re
def analyze_content(body):
suspicious_links = re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', body)
if suspicious_links:
return "Найденыподозрительныессылки"
else:
return "Содержимое не содержит подозрительных элементов"
```

2. Модуль классификации сообщений:

Этот модуль отвечает за классификацию сообщений как фишинговых или нормальных на основе данных, полученных из модуля загрузки и анализа сообщений. Он включает следующие компоненты:

- Функция классификации: Эта функция принимает результаты анализа заголовков и содержимого сообщения и определяет его статус как фишинговое или нормальное. Примеркода:

```python
def classify_message(headers
_analysis, content_analysis):
if headers_analysis == "Подозрительныйотправитель" or content_analysis == "Найденыподозрительныессылки":
return "Фишинговое сообщение"
else:
return "Нормальное сообщение"
```

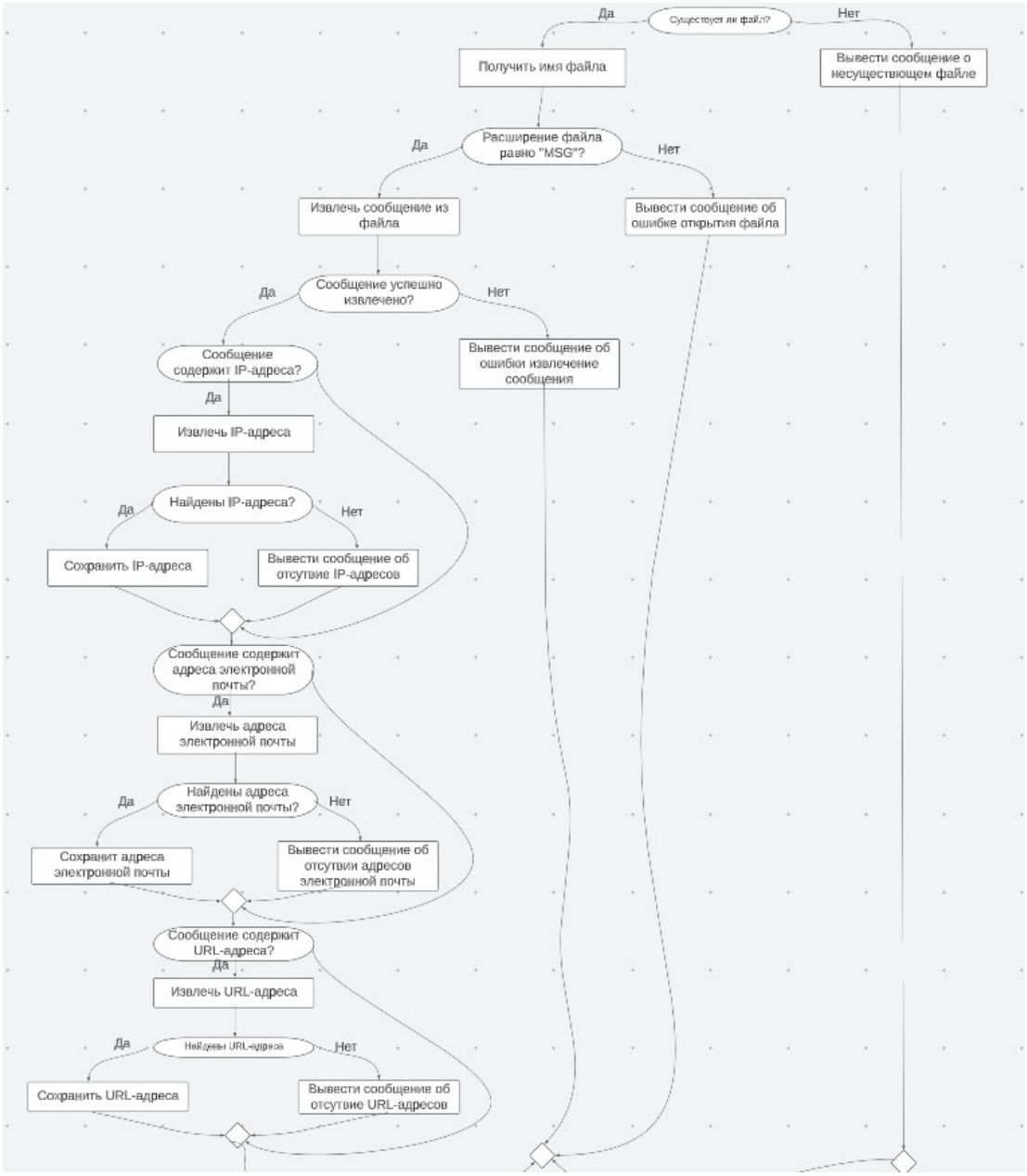


Рисунок 5 – Диаграмма активностей

3. Модуль вывода результатов:

Этот модуль отвечает за вывод результатов анализа и классификации сообщений. Он может включать следующие компоненты:

- Функция вывода: Эта функция принимает результаты классификации сообщения и выводит их на экран или сохраняет в файл. Примеркода:

```python
def display_results(message, classification):
print("Результатыанализасообщения:")
print("Сообщение:", message)
print("Классификация:", classification)
```

```

4. Основная программа:

В основной программе происходит последовательный вызов модулей загрузки и анализа сообщений, классификации и вывода результатов. Примеркода:

```python
def main():
file_path = "path/to/message.msg"  #Путькфайлусообщения
# Загрузкаианализсообщения
message = load_message(file_path)
headers_analysis = analyze_headers(message['headers'])
content_analysis = analyze_content(message['body'])
# Классификациясообщения
classification = classify_message(headers_analysis, content_analysis)
# Выводрезультатов
display_results(message, classification)
if __name__ == "__main__":
main()
```

В этом примере основная программа загружает сообщение, анализирует его заголовки и содержимое, классифицирует его и выводит результаты на экран.

Такая структура программы обеспечивает последовательную обработку сообщений, анализ и классификацию на основе заданных алгоритмов и методик. Каждый модуль выполняет свою функцию, а основная программа координирует их работу, обеспечивая обработку и вывод результатов для каждого сообщения.

3.2 Реализация алгоритмов анализа сообщений

Для реализации алгоритмов анализа сообщений в разработанном программном решении на Python мы используем различные методы и инструменты для анализа заголовков и содержимого сообщений. Давайте рассмотрим каждый алгоритм подробно:

1. Анализ заголовков сообщений:

Анализ заголовков сообщений выполняется с целью проверки подлинности отправителя и других признаков, которые могут указывать на фишинговое сообщение. Для этого мы реализуем следующий алгоритм:

- Извлечение адреса электронной почты отправителя из заголовков сообщения.
- Разделение адреса на имя пользователя и домен.
- Проверка домена отправителя на наличие в списке доверенных доменов.
```

Вот пример кода, демонстрирующего реализацию этого алгоритма:

```python
def analyze_headers(headers):
sender_email = headers['From']
sender_domain = sender_email.split('@')[1]
known_domains = ['example.com', 'company.com', 'organization.org']
if sender_domain not in known_domains:
return "Подозрительный отправитель"
else:
return "Доверенный отправитель"
```

В этом коде мы извлекаем адрес отправителя из заголовков и сравниваем его домен с предопределенным списком доверенных доменов. Если домен отправителя не находится в списке доверенных, мы классифицируем его как "Подозрительный отправитель".

2. Анализ содержимого сообщений:

Анализ содержимого сообщений выполняется для поиска подозрительных ссылок, фраз или других признаков, которые могут указывать на фишинговое сообщение. Для этого мы реализуем следующий алгоритм:

- Поиск URL-адресов в тексте сообщения с использованием регулярных выражений или специализированных библиотек для анализа текста.

- Проверка найденных URL-адресов на наличие подозрительных характеристик, таких как неизвестный домен, использование замаскированных символов или наличие других признаков фишинга.

Вот пример кода, демонстрирующего реализацию этого алгоритма:

```python
import re
def analyze_content(body):
suspicious_links = re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', body)
if suspicious_links:
return "Найденыподозрительныессылки"
else:
return "Подозрительные ссылки не обнаружены"
```

В этом коде мы используем регулярное выражение для поиска URL-адресов в тексте сообщения. Затем мы проверяем найденные ссылки на наличие подозрительных признаков. Если обнаружены подозрительные ссылки, мы классифицируем сообщение соответственно.

Таким образом, реализация алгоритмов анализа сообщений в программном решении на Python позволяет тщательно анализировать заголовки и содержимое сообщений и определять их статус как фишинговые или нормальные на основе найденных признаков.

Сформулировав определение термину «политика информационной безопасности», следует рассмотреть для чего же нужна политика информационной безопасности в организации и какие цели с помощью неё можно достичь.

Цели политики информационной безопасности относятся к одной или нескольким из следующих категорий:

- защита ресурсов; а
- утентификация;
- авторизация;
- целостность;
- конфиденциальность;
- аудит безопасности.

Таким образом, можно сделать вывод, что политика информационной безопасности является неотъемлемым элементом любой организации. Задействование политики ИБ может существенно увеличить уровень защити информации, что в свою очередь влечет снижение риска финансовых и репутационных потерь.

В области информационной безопасности компании, которые занимаются разработкой ПО проводят свои исследования. Данные исследования полезны для изучения специалистам по информационной безопасности, для того чтобы спрогнозировать тип и характер угроз. Согласно отчёту портала ptsecurity в первом квартале 2019 года наиболее распространёнными средствами атаки на финансовые организации стало использование ВПО и социальная инженерия или

«фишинг», именно «фишинг» направлен на задействование сотрудников организации. На рисунке 6 представлена диаграмма методов атак на финансовые организации.[33, 36]

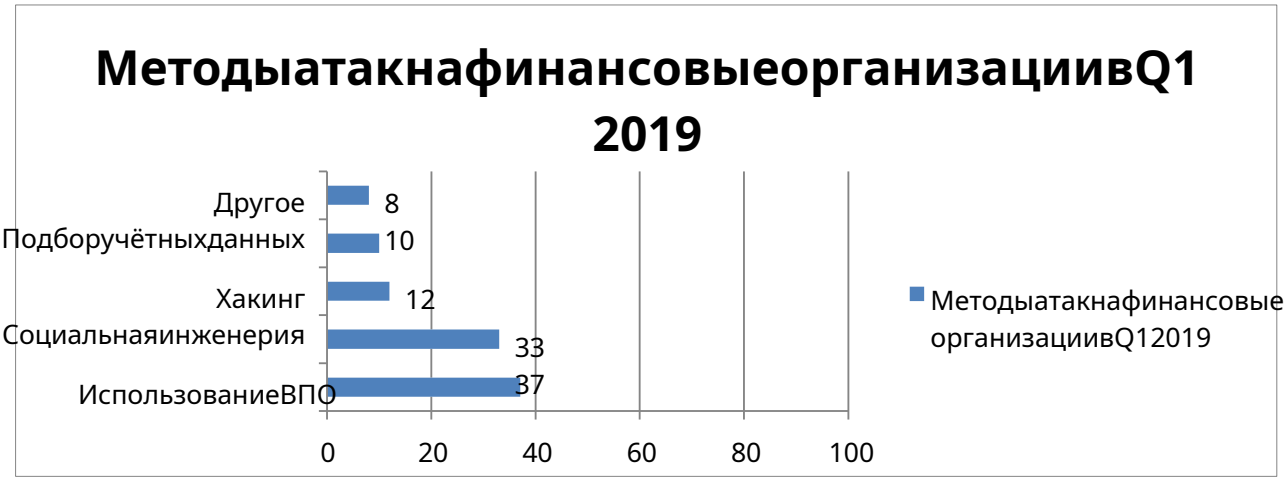


Рисунок 6 – Методы атак на финансовые организации в Q1 2019

Если учесть, что средства защиты развиваются вместе с угрозами, то можно предположить, что способы и задействованные средства при кибератаках претерпят существенные изменения. Наиболее вероятный метод, это осуществить атаку с помощью ничего не подозревающего сотрудника, например, чтобы он запустил вредоносный исполняемый файл. Такой вид атак классифицируется как

«социальная инженерия», в повседневной жизни такие атаки называются

«фишинг» или «инсайдерские».

Как видно из диаграммы, такой вид атак почти стал вровень по популярности с ВПО. Основная опасность фишинга заключается в том, что его трудно предотвратить. Подобный вид атак, всё ещё остаётся наиболее опасным для финансовых организаций, так как реализованная угроза с помощью сотрудника, может повлечь серьёзные последствия. Защита организации изнутри всё ещё является слабым местом многих организаций.

В подтверждение слов о серьёзности внутренней угрозы. Портал anti-malware провёл исследование о характере внутренних угроз. В результате этого исследования была составлена диаграмма.

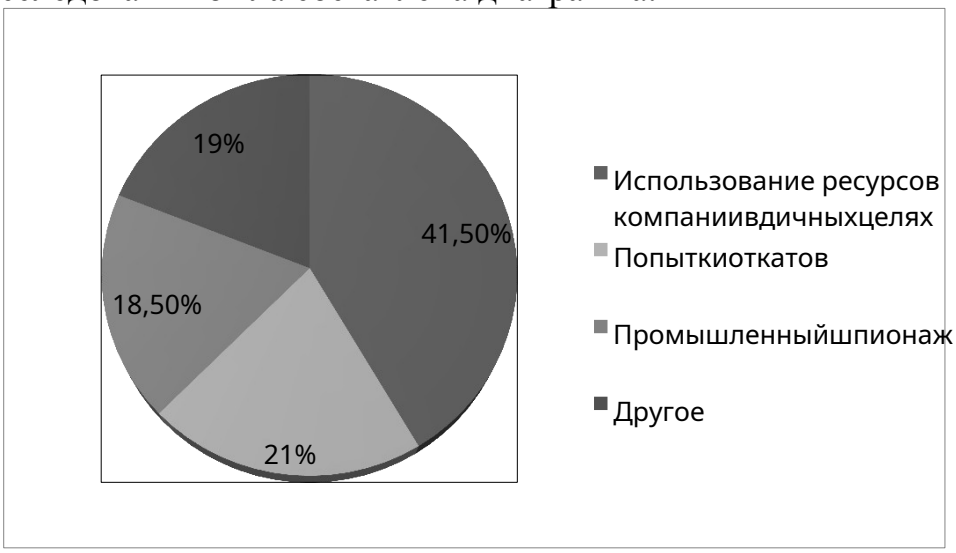


Рисунок 7 – Диаграмма угроз

3.3 Обработка исключительных ситуаций и ошибок в программе

В разрабатываемом программном решении на Python для обработки исключительных ситуаций и ошибок мы используем механизм исключений и обработку исключений. Это позволяет корректно обрабатывать возможные ошибки и ситуации, которые могут возникнуть в процессе работы программы. Давайте рассмотрим примеры обработки исключительных ситуаций:

1. Обработка ошибок загрузки сообщения:

Если при загрузке сообщения возникает ошибка, например, файл сообщения не найден или некорректный формат файла, мы должны предусмотреть соответствующую обработку ошибок. Вотпримеркода:

```python
def load_message(file_path):
try:
# Код загрузки сообщения из файла
# ...
return message
except FileNotFoundError:
print("Ошибка: Файлсообщенияненайден.")
return None
exceptInvalidFileFormatError:
print("Ошибка: Некорректныйформатфайласообщения.")
returnNone
```

В этом коде мы используем блок `try-except` для перехвата исключений, которые могут возникнуть при загрузке сообщения. Если возникает ошибка `FileNotFoundError`, мы выводим сообщение об ошибке, указывающее, что файл сообщения не найден. Если возникает ошибка `InvalidFileFormatError`, мы выводим сообщение о некорректном формате файла.

2. Обработка ошибок анализа сообщений:

В процессе анализа заголовков и содержимого сообщений также могут возникать ошибки, например, если структура сообщения не соответствует ожидаемому формату или отсутствуют необходимые данные. Вот пример обработки ошибок в алгоритме анализа заголовков:

```python
defanalyze_headers(headers):
try:
# Коданализазаголовков
# ...
return analysis_result
exceptMissingDataError:
print("Ошибка: Отсутствуют необходимые данные в заголовках сообщения.")
return None
except InvalidHeaderFormatError:
print("Ошибка: Некорректныйформатзаголовковсообщения.")
returnNone
```

В этом коде мы использовали блок `try-except` для перехвата возможных исключений при анализе заголовков. Если возникает ошибка `MissingDataError`, мы выводим сообщение о отсутствии необходимых

данных в заголовках. Если возникает ошибка `InvalidHeaderFormatError`, мы выводим сообщение о некорректном формате заголовков.

Таким образом, обработка исключительных ситуаций и ошибок в программе позволяет корректно реагировать на возможные проблемы и уведомлять пользователя о них, обеспечивая более стабильное и надежное функционирование программного решения.

3.4 Подготовка тестовых наборов данных

Для тестирования и валидации разработанной методики и программного решения мы подготавливаем тестовые наборы данных, которые содержат разнообразные примеры электронных сообщений. Эти наборы данных помогут нам проверить работу алгоритмов анализа и обработки сообщений, а также оценить эффективность методики защиты от целевого фишинга.

При подготовке тестовых наборов данных мы учитываем следующие аспекты:

1. Разнообразие типов сообщений: Мы включаем в тестовые наборы данных разнообразные типы сообщений, такие как обычные письма, рекламные сообщения, социальные сети, банковские уведомления и т.д. Это поможет нам проверить, насколько хорошо методика справляется с различными видами фишинга.

2. Поддельные фишинговые сообщения: Мы включаем в тестовые наборы данных фишинговые сообщения, которые созданы специально для тестирования системы. Это поможет нам оценить эффективность методики в распознавании и обнаружении таких сообщений.

3. Реальные примеры фишинговых сообщений: Мы также включаем реальные примеры фишинговых сообщений из открытых источников или баз данных фишинговых атак. Это поможет нам проверить, насколько хорошо методика справляется с актуальными угрозами.

4. Обычные и легитимные сообщения: В тестовых наборах данных также должны присутствовать обычные и легитимные сообщения, чтобы оценить, насколько низка вероятность ложных срабатываний методики и программного решения.

Кроме того, важно убедиться, что тестовые наборы данных достаточно объемные и разнообразные для полноценного покрытия возможных сценариев и ситуаций.

Подготовка тестовых наборов данных требует систематического подхода и ручной работы. Мы можем создать различные папки с файлами сообщений разных типов, организовать структуру каталогов, указывающую на содержание и свойства каждого сообщения, и разделить наборы данных на обучающие, тестовые и валидационные наборы.

Важно также учитывать правовые и этические аспекты при использовании реальных фишинговых сообщений и соблюдать соответствующие правила и нормы.

3.5 Анализ результатов тестирования и корректировка программного решения

После проведения тестирования разработанной методики и программного решения, мы проводим анализ полученных результатов с целью оценки их эффективности и выявления необходимых корректировок. Этот анализ позволяет нам оценить успешность методики в борьбе с целевым фишингом, а также определить качество работы программного решения. Важно учитывать следующие факторы:

1. Эффективность обнаружения: Оценка способности методики и программного решения обнаруживать целевой фишинг. Мы анализируем процент успешно обнаруженных фишинговых атак относительно общего числа попыток.
2. Ложноположительные срабатывания: Оценка количества ложноположительных результатов, то есть случаев, когда методика неправильно идентифицирует легитимные сообщения как фишинговые. Слишком высокий уровень ложноположительных срабатываний может негативно сказаться на удобстве использования и доверии к методике.
3. Реакция на новые угрозы: Оценка гибкости методики и ее способности адаптироваться к новым видам и методам целевого фишинга. Мы анализируем, насколько быстро методика обнаруживает и реагирует на новые угрозы, а также насколько успешно она предотвращает атаки.
4. Уведомление пользователей: Оценка эффективности механизма уведомления пользователей о потенциальных фишинговых атаках. Мы анализируем, насколько пользователи реагируют на предупреждающие сообщения и принимают меры для защиты своих данных.

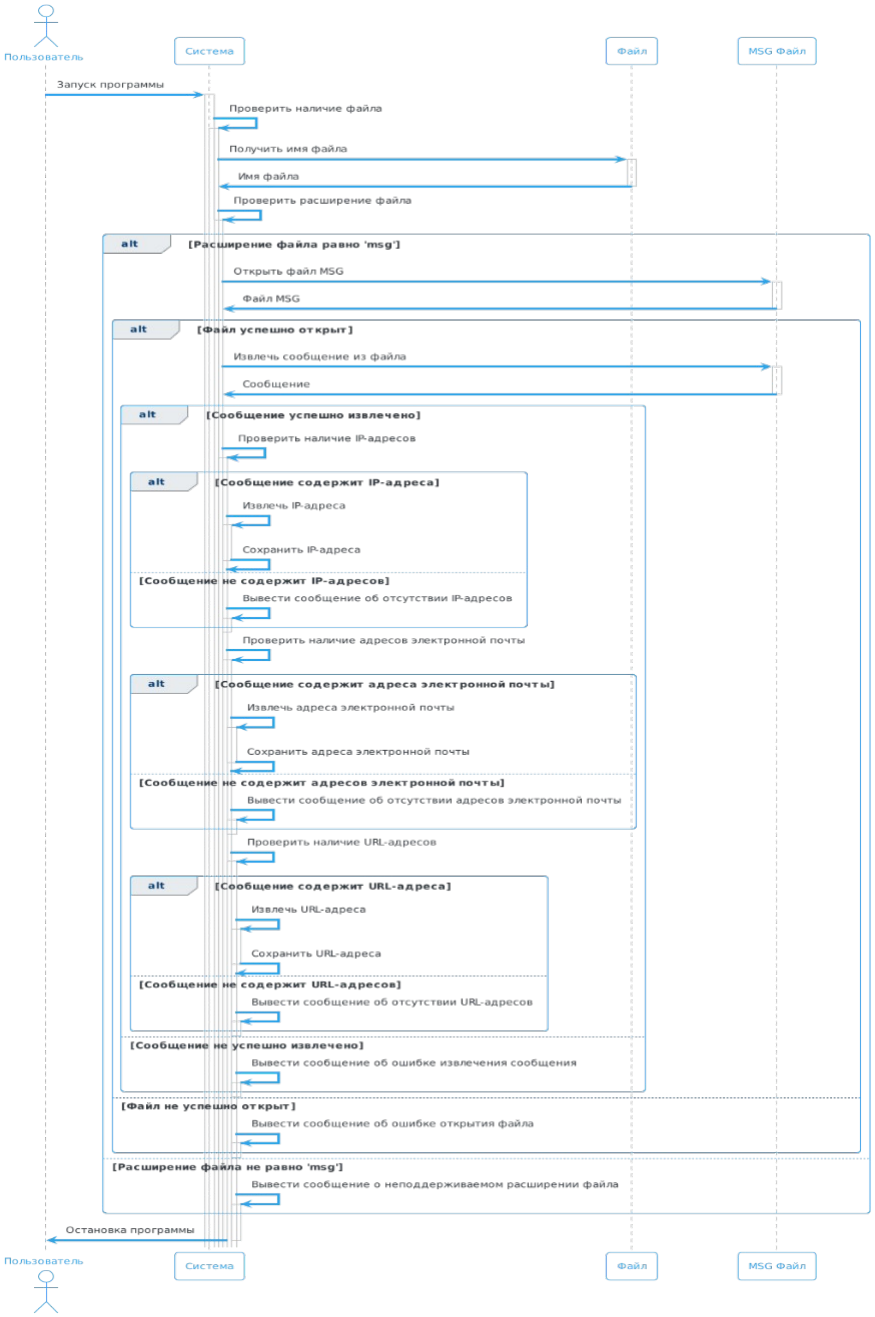


Рисунок 8 –Диаграмма последовательностей

В процессе анализа результатов тестирования мы выполняем следующие шаги:

1. Оценка точности и полноты: Мы анализируем результаты классификации сообщений и определяем, насколько точно методика и программное решение распознают и классифицируют фишинговые и нормальные сообщения. Мы сравниваем результаты с исходными метками сообщений из тестовых наборов данных.

2. Оценка ложных срабатываний и пропусков: Мы анализируем количество ложных срабатываний, когда методика ошибочно классифицируетнормальные сообщения как фишинговые, и пропусков, когда фишинговые сообщения не распознаются. Это помогает нам определить границы и уровень надежности методики.

3. Идентификация проблемных сценариев: Мы анализируем результаты для выявления типичных проблемных сценариев, в которых методика и программное решение не справляются с обнаружением или классификацией фишинговых сообщений. Это позволяет нам идентифицировать слабые места и причины недостаточной эффективности.

4. Корректировка программного решения: На основе анализа результатов тестирования мы вносим соответствующие корректировки в программное решение. Это может включать изменение алгоритмов анализа, настройку параметров классификации или добавление новых функций для улучшения точности и полноты методики.

После внесения корректировок мы повторно тестируем методику и программное решение для проверки улучшений и повышения их эффективности. Этот процесс итеративно повторяется до достижения удовлетворительных результатов и соответствия требованиям, сформулированным в начале работы.

Важно также документировать результаты анализа, внесенные корректировки и основные выводы, чтобы обеспечить прозрачность и возможность последующей проверки и повторного использования разработанной методики и программного решения.

3.6 Основные функций программы

Для начала работы программы мы заходим в свою почту и находим подозрительное письмо, которое нам отправили. Это может быть спам, рассылка или фишинговое письмо. Письмо должно быть в формате .msg для корректной работы. Вот пример письма, который мы возьмём для разбора:

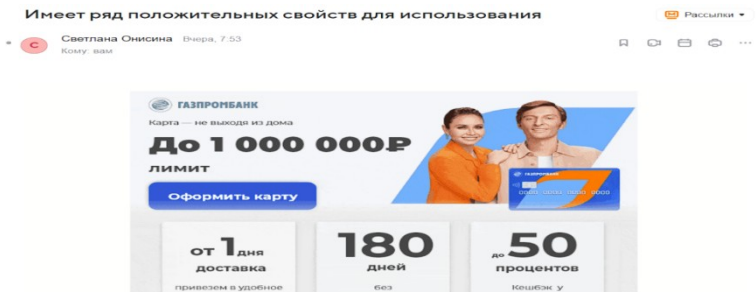


Рисунок 9 – Пример письма для анализа

Для начала анализа, мы скачаем данное письмо на компьютер. Затем, для удобства работы в программе, мы переименуем его. При скачивании письма, оно обязательно должно иметь расширение **.msg,** так как программа работает именно с этим расширением.

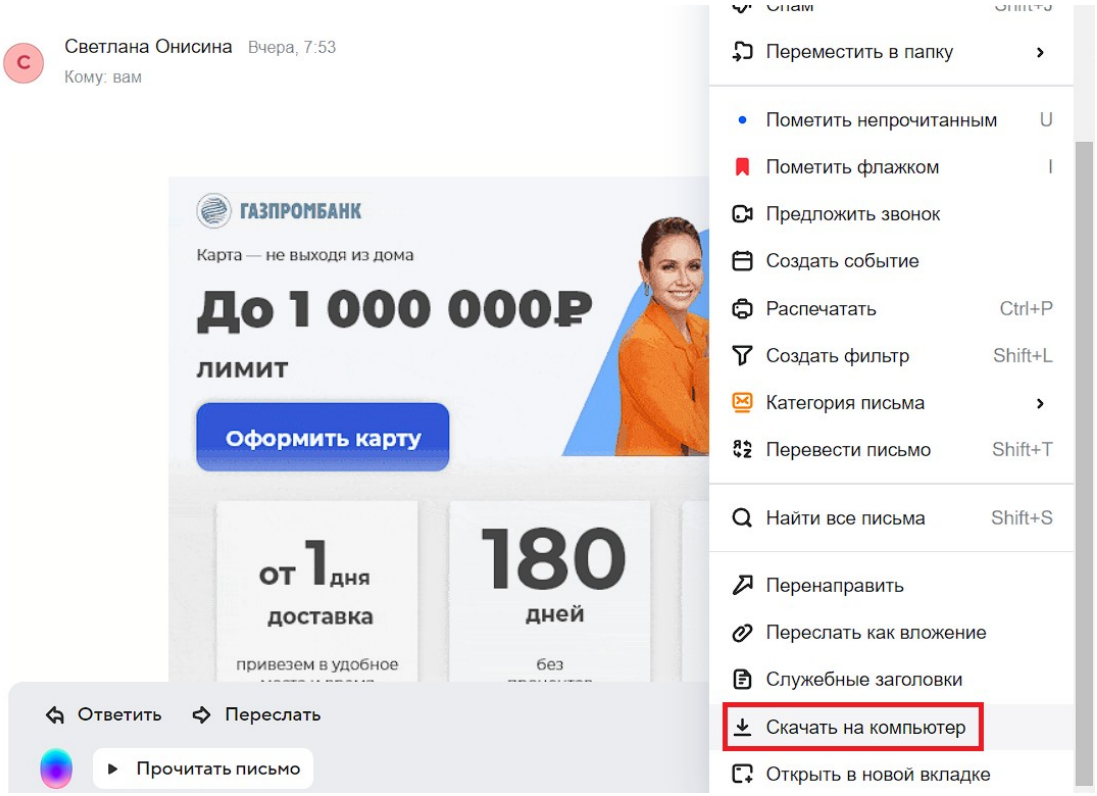


Рисунок1 0 – Расположение кнопки для скачивание письма

Расположение кнопки для скачивания письма может варьироваться в зависимости от почтового клиента или программы, которую вы используете. Обычно кнопка для скачивания письма находится вверху или внизу самого письма, возле других доступных действий, таких как ответить, переслать или удалить. Она может быть представлена в виде значка документа или стрелки, указывающей на скачивание. Также вы можете найти опцию скачивания письма в контекстном меню, вызываемом правым кликом на самом письме.

В качестве примера, используется почтовая служба**mail.ru**, на рисунки выше приведён пример кнопки "Скачать на компьютер", чтобы скачать данное сообщение в формате .**msg.** После этого вы можете переименовать его в "**015.eml**" для большего удобства. Затем вы можете перекинуть данный файл в вашу программу, а именно в сам код, для дальнейшего анализа.

Путем скачивания и переименования файла письма вы готовите его для обработки и анализапрограммой. Это важный шаг, посколькунужно обеспечить корректную обработку и правильное распознавание формата письма программой. После перемещения файла в программу вы можете продолжить с анализом содержимого письма и провести необходимые

проверки на наличие фишинговых элементов или других потенциальных угроз.

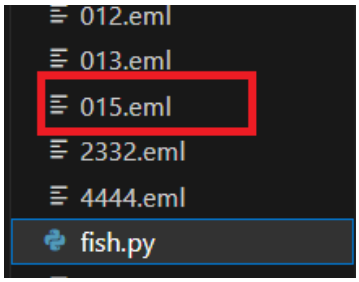


Рисунок 11 – Расположение скаченного письма

Расположение скаченного письма зависит от настроек и выбранного места сохранения на вашем компьютере. Обычно скаченные письма сохраняются в папке "Загрузки" или в папке, которую вы указали в качестве места сохранения. Вы также можете выбрать конкретную папку для сохранения писем при скачивании.

После того, как мы перекинули файл письма в нашу программу для начала анализа, мы должны написать в терминале команду "**python fish.py 015.eml**". Данная команда позволяет запустить анализ сообщения. Ниже приведен пример выполнения этой команды:

Рисунок 12 – Выполнения команды запуска приложения

При выполнении команды "**python fish.py 015.eml**" программа загрузит файл письма с именем "**015.eml**" и начнет его анализ. Анализ может включать проверку отправителя, содержимого письма на наличие признаков фишинга, анализ вложений или ссылок, а также оценку уровня риска. В результате анализа программа может сформировать отчет с результатами, указывающим, является ли письмо подозрительным или потенциально опасным.

После выполнения команды запуска приложения, оно начинает анализировать содержимое скачанного письма и выводит конечные результаты проверки в виде разделов. Эти разделы включают, например, раздел с подозрительностью и раздел с заголовками письма.

В разделе подозрительности программное решение анализирует различные аспекты письма, чтобы определить, насколько оно подозрительное или потенциально опасное. Это может включать проверку отправителя, содержимого письма, вложений, ссылок и других признаков фишинга или

мошенничества. Результаты этой проверки представлены в удобной форме, позволяющей вам легко определить уровень подозрительности письма.

В разделе заголовков письма представлены все заголовки, которые были прописаны в сообщении при его отправке. Это может включать информацию о отправителе, получателе. Вывод этих заголовков позволяет вам получить полную информацию о письме и использовать ее для дополнительного анализа или принятия решений.

Общие результаты анализа письма предоставляют вам информацию о его подозрительности, наличии фишинговых элементов или потенциальных угроз. Это помогает вам оценить, насколько безопасно взаимодействовать с письмом и принять соответствующие меры, например, удалить его, пометить как спам или обратиться к дополнительным инструментам или специалистам для проверки.

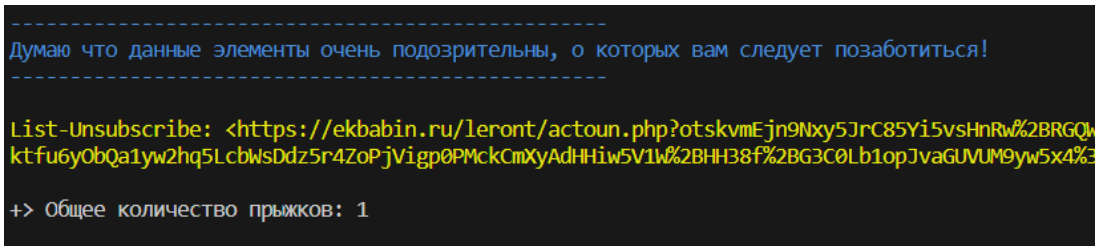


Рисунок 13 – Подозрительные элементы в письма

В первом разделе проверки выводятся подозрительные элементы письма, которые могут быть обнаружены программой. На рисунке приведен пример, в котором видно наличие подозрительного URL-адреса. В данном случае URL-адрес "**List-Unsubscribe**" помечается как подозрительный.

"**List-Unsubscribe**" является ссылкой на отписку от рассылки писем. Если нажать на данную ссылку, можно отписаться от получения дальнейших писем этой рассылки. Это важно для пользователей, которые не хотят продолжать получать такие письма.

Ниже ссылки "Общее количество прыжков" указывает на значение, сколько раз программа прошлась по нашему письму и проанализировала все его данные. Это показывает, что программа выполнила необходимый анализ, просмотрев различные аспекты письма для определения его подозрительности или безопасности.

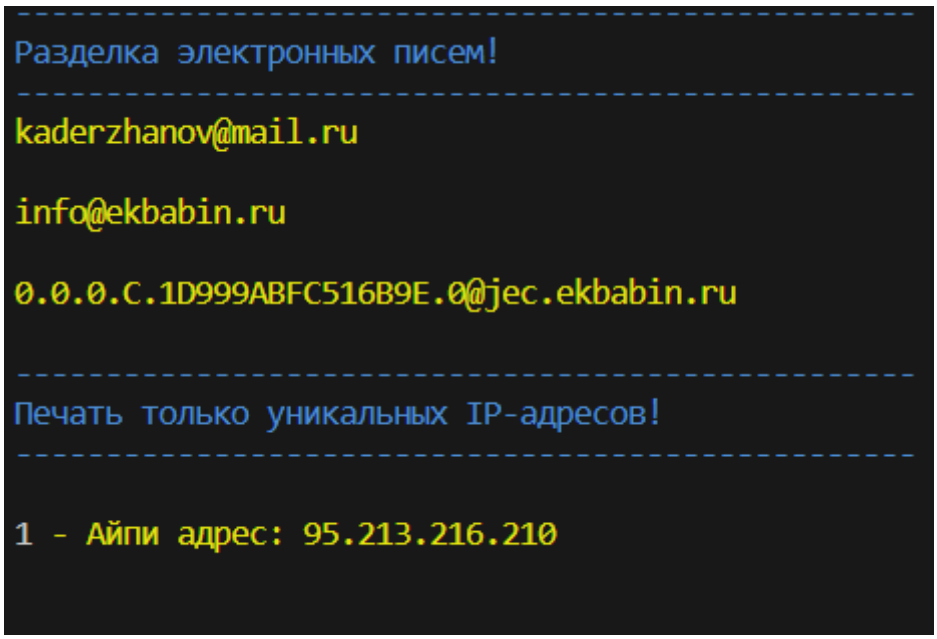


Рисунок 14 – Вывод разделки электронных писем и уникальных IP– адресов

На данном рисунке мы видим две разделки. Давайте начнем с разделки "Письма". В этой разделке указано три электронных адреса, которые были использованы при отправке нам данного сообщения.
Первый адрес **kaderzhanov@mail.ru** является нашим адресом, то есть адресом, на который было отправлено письмо.
Второй адрес **info@ekbabin.ru** является адресом отправителя письма. Это электронный адрес, указанный в поле "От" или "Отправитель".
Третий адрес **0.0.0.C.1D999ABFC516B9E.0@jec.ekbabin.ru** является идентификатором почты отправителя. Это уникальный идентификатор, который может быть использован для идентификации отправителя.
В разделке "Печать только уникальных IP-адресов!" указан уникальный IP-адрес отправителя письма. Это означает, что программа отображает только уникальные IP-адреса, которые были связаны с отправителем письма.

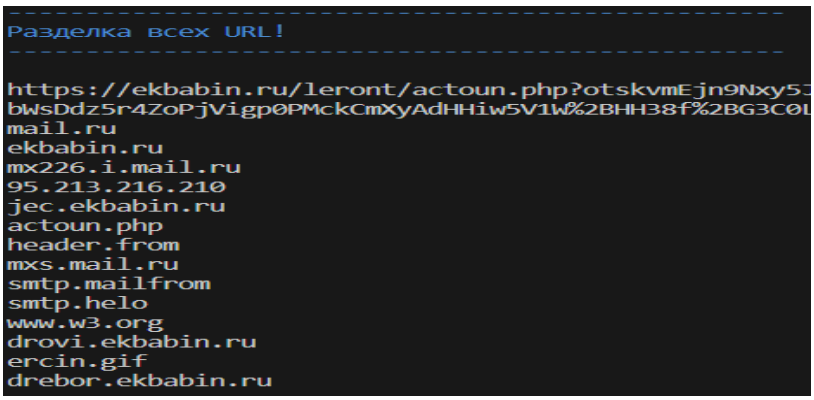


Рисунок 15 – Разделка всех URL найденных в письме при анализе

В данной разделке мы видим список добавленных URL при отправке сообщения. Здесь перечислены все сайты, которые были использованы при получении письма на нашу почту. Эти URL-адреса могут представлять ссылки, вложения или другие ресурсы, связанные с письмом. Анализ этих URL-адресов может помочь определить наличие подозрительных или потенциально опасных ссылок в письме.

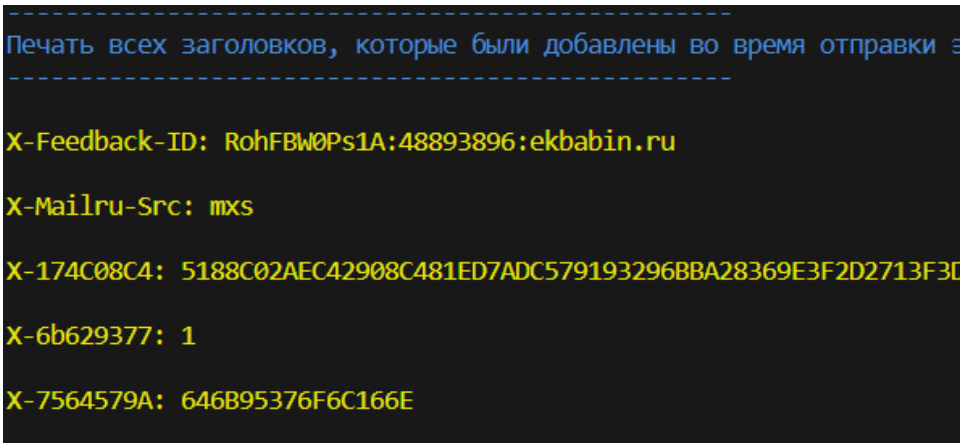


Рисунок 3.16–В данном выводе представлены все заголовки писем, отправленных во время отправки

На данном рисунке выводятся все заголовки, которые были добавлены во время отправки сообщения. Заголовки содержат различные метаданные и информацию о письме, такие как отправитель, получатель, тема, дата и другие параметры. Эти заголовки играют важную роль при анализе и идентификации сообщения. Они помогают определить подлинность и происхождение письма, а также могут содержать полезную информацию для дальнейшего анализа и обработки.

Заголовки письма могут содержать различные обозначения и информацию. Вот некоторые распространенные обозначения заголовков:

Заголовок "**X-Feedback-ID**" представляет собой специальный заголовок, который может быть добавлен в письмо для целей обратной связи или идентификации. В данном случае, значение заголовка "**RohFBW0Ps1A:48893896:ekbabin.ru**" может представлять уникальный идентификатор или код, связанный с обратной связью или отслеживанием письма.

Заголовок "**X-Mailru-Src**" указывает на источник отправки письма и может содержать информацию о том, каким образом письмо было обработано и передано почтовой системой **Mail.ru.** В данном случае, значение "mxs" указывает на использование почтового сервера (**Mail Exchange Server**) в качестве источника отправки.

Заголовок "**X-174C08C4**" представляет собой пользовательский или системный заголовок, который может быть уникальным идентификатором

или кодом, связанным с обработкой или маркировкой письма. В данном случае, значение "**5188C02AEC42908C481ED7ADC579193296BBA28369E3F2D2713F3D5F7D406D31BCF678C7329BA986**" представляет собой конкретное значение или идентификатор для данного письма.

Заголовок "**X-6b629377**" является пользовательским заголовком, который может быть использован для различных целей в обработке и маркировке писем. В данном случае, значение "**1**" указывает на конкретное значение или состояние, связанное с письмом.

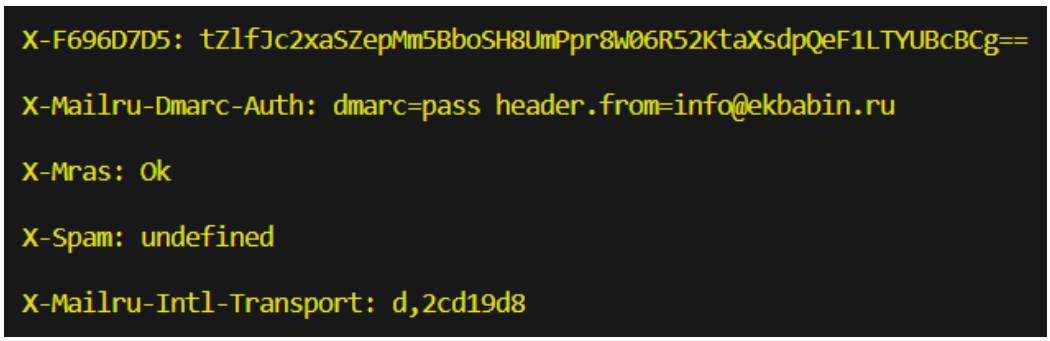


Рисунок 17 – Продолжение найденных заголовков при анализе

Заголовок "**X-Mailru-Dmarc-Auth**" указывает на результат аутентификации **DMARC** (**Domain-based Message Authentication, Reporting, andConformance**) для заголовка "From" письма.

Значение "dmarc=pass" указывает на успешную проверку аутентификации DMARC. Это означает, что домен "ekbabin.ru", указанный в заголовке "**From**" письма (info@ekbabin.ru), прошел проверку **DMARC** и соответствует политикам аутентификации, установленным для этого домена.

DMARC- это стандарт аутентификации писем, который позволяет проверять подлинность отправителя и защищать от фальсификации адресов электронной почты. Проверка **DMARC** включает в себя проверку **SPF** (**Sender Policy Framework)** и **DKIM** (**DomainKeysIdentified Mail**), чтобы убедиться, что письмо отправлено от имени действительного домена и не было поддельным или измененным в пути доставки.

В данном случае, результат "**dmarc=pass**" указывает на то, что письмо было отправлено от домена "**ekbabin.ru**" и прошло успешную аутентификацию по правилам **DMARC**. Это помогает установить доверие в источник письма и подтверждает, что адрес отправителя (info@ekbabin.ru) действительный.

Заголовок "**X-Mras**" указывает на результат проверки письма анти-спам системой **MRAS** (**Mail.Ru Anti-Spam**). Значение "**Ok**" в данном случае указывает на то, что письмо успешно прошло проверку и не было классифицировано как спам.

Заголовок "**X-Spam**" указывает на результат проверки спама или фильтрации спама для данного письма. Однако, значение "**undefined**" в этом заголовке может означать, что нет явного указания на результат проверки спама.

Обычно, если письмо было отфильтровано как спам, заголовок "**X-Spam**" будет содержать дополнительные сведения о результатах проверки, такие как метки, оценки или значения, указывающие на вероятность того, что письмо является спамом. Однако, в данном случае значение "**undefined**" может указывать на то, что информация о результате проверки спама не была явно предоставлена или не была обработана.

Заголовок "**X-Mailru-Intl-Transport**" указывает на информацию о маршрутизации или транспортировке письма, связанную с почтовой службой **Mail.ru**. Значение "**d,2cd19d8**" в данном заголовке представляет собой идентификатор или код, используемый внутренними системами Mail.ru для обработки и пересылки писем.

В результате анализа мы получаем доступ ко всей информации, содержащейся в письме. Мы извлекаем все данные, которые были в нем записаны, а также проводим полный анализ содержимого, включая скрытые данные, уникальные IP-адреса и электронные адреса, использованные при отправке и получении письма. Мы также анализируем все заголовки, используемые в письме. Благодаря этой информации мы способны проанализировать данные и определить, является ли сообщение безопасным или нет.

Дополнительно, мы также проводим анализ следующих аспектов:

Аутентичность отправителя: Оценка достоверности и подлинности отправителя письма. Мы проверяем подлинность электронной почты, связанной с отправителем, и анализируем другие факторы, указывающие на возможное подделывание или фальсификацию идентификации отправителя.

Форматирование и структура письма: Анализ стиля, грамматики и форматирования письма для выявления необычных или подозрительных элементов. Мы исследуем наличие опечаток, грамматических ошибок, несоответствия стиля и других признаков, которые могут указывать на потенциальную фишинговую атаку.

Ссылки и вложения: Проверка ссылок и вложений, содержащихся в письме. Мы анализируем URL-адреса, указанные в сообщении, чтобы определить, являются ли они подозрительными или ведут на фальшивые или опасные сайты. Также проводится анализ вложенных файлов на наличие вредоносного программного обеспечения или других угроз.

История и контекст: Исследование предыдущей истории взаимодействия с отправителем или связанными с ним сущностями. Мы анализируем предыдущие письма и события, связанные с отправителем, чтобы определить, есть ли какие-либо подозрительные или необычные образцы или поведение.

Общая оценка риска: На основе всех проведенных анализов мы осуществляем общую оценку риска, связанного с письмом. Это позволяет нам определить, является ли сообщение потенциальной фишинговой атакой или безопасным обращением.

Все эти аспекты анализа помогают нам принимать информированные решения относительно безопасности письма и противодействовать целевому фишингу.

4 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

В данном разделе проводится анализ рабочих условий, с целью рассмотрения мероприятий по обеспечению безопасности и улучшению условий труда в процессе проектирования и обеспечения безопасности сети.

Для достижения этой цели в рамках данной дипломной работы предлагается разработать программу на языке python для предотвращения фишинга. Эта программа позволит пользователям проводить анализ полученного электронного письма в формате .emlи.msg.Программа на языке python будет разработана с использованием последних технологий и современных подходов к защите данных.

Пользователи смогут узнать всю информацию об отправителе данного письма и возможные рекомендации.

В ходе анализа условий труда будут учтены такие факторы, эргономика рабочего места, психологическая обстановка и другие аспекты, влияющие на безопасность и комфортность работы.

Эргономические травмы. Офисные работники чаще всего получают травмы позвоночника, зрения и т.п. из-за неправильной осанки во время работы, постоянный сидячий рабочий процесс, и без постоянных разминок.

Рекомендации по управлению и безопасности труда изображены на рисунке 4:

- расположить кресло, клавиатуру и монитор на одной линии с телом;
- сохранять расслабленную нейтральную позу;
- сидеть прямо, регулируя стул, чтобы обеспечить твердую поддержку спины;
- пусть ваши руки свободно висят на плечах;
- во время набора текста держать локти под углом 90 градусов;
- использовать регулируемый лоток для клавиатуры, чтобы расположить клавиатуру и мышь на удобной высоте (обычно ниже поверхности стола). Поместить мышь рядом с клавиатурой и держать ее как можно ближе к телу, чтобы избежать ее попадания;
- регулировать высоту стула так, чтобы ноги прочно стояли на земле.

Рисунок 18 – Эргономика рабочего места

Психологическая обстановка

Психологическая обстановка на рабочем месте должна быть поддерживающей и стимулирующей для работников. Вот некоторые основные аспекты, которые способствуют здоровой психологической обстановке:

1. Коммуникация: Важно, чтобы сотрудники могли свободно общаться и обмениваться информацией. Открытость и прозрачность в коммуникации помогают снизить стресс и создают ощущение принадлежности к команде.

2. Поддержка и уважение: Работники должны чувствовать, что их мнение ценится, и получать поддержку со стороны коллег и руководства. Уважительное отношение к различным точкам зрения и индивидуальным потребностям помогает создать благоприятную рабочую атмосферу.

3. Работа-жизнь баланс: Предоставление возможностей для соблюдения рабочего времени и личной жизни помогает сотрудникам справляться с нагрузкой и избегать переутомления. Гибкий график работы и поддержка при совмещении личных и профессиональных обязанностей способствуют психологическому благополучию.

4. Поддержка развития: Предоставление возможностей для профессионального роста и развития помогает сотрудникам ощущать себя ценными и дает им перспективы для карьерного развития. Обучение, тренинги и возможности повышения квалификации способствуют мотивации и удовлетворенности работой.

5. Управление стрессом: Работодатели могут предоставить ресурсы и программы по управлению стрессом, помочь работникам находить баланс и развивать навыки эмоционального управления. Предоставление психологической поддержки и консультирование также могут быть полезными для справления с работоспособностью и эмоциональными вызовами.

Важно отметить, что каждая организация может иметь свои особенности и потребности в создании психологически благоприятной обстановки на рабочем месте. Это требует индивидуального подхода и адаптации к конкретной ситуации

Астенопия (быстрая утомляемость глаз). При длительной работе за компьютером (более 4 часов в сутки) наблюдается нарушение аппомодации и цветового восприятия. Астенопию вызывают:

- мерцание экрана;
- плохая освещенность как рабочего места, так и экрана, высокая резкость, наличие бликов и отражений, неоптимальное соотношение яркости и контрастности. Все это влияет не только на глаза, но и на мозг, который анализирует изображение и дает команду рукам.

4.1 Рекомендуемое помещение в котором будет вестись разработка

Давайте рассмотрим рекомендуемые параметры для помещения, предназначенного для разработки:

Тип помещения: офисное помещение.

- Размеры рабочего помещения:
- Длина: 6 метра.
- Ширина: 4 метров.
- Высота: 3 метра.
- Площадь: 24 м2

Присутствие остекления в помещении.

- Искусственное освещение с использованием 2 светильников, каждый из которых содержит 2 люминесцентные лампы.

Вид работы: разработка управленческого сайта приложением.

- Количество рабочих мест: 3.
- Категория работ: легкая.

4.2 Оборудование и эскиз рабочего места

Схема помещения представлен на рисунке 5

Наше помещение, а это стены, потолок, паркет и мебель выполнено строго в белых тонах для лучшего комфорта работы, а высота потолков составляет 3 метра.

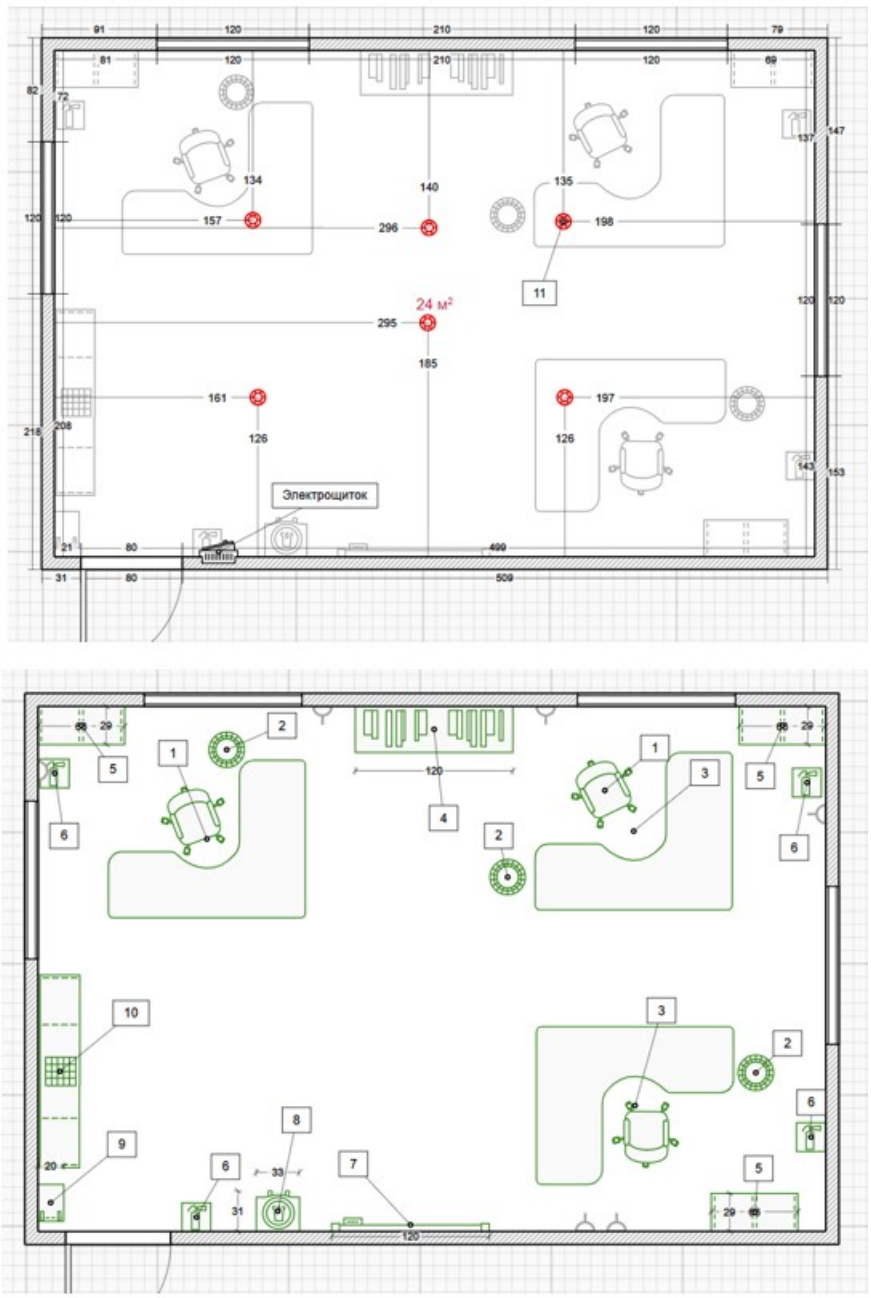


Рисунок 19 – Схема помещения

Характеристика помещения:

1 – Кресло офисное;
2 – Урна для мусора;
3 – Рабочий стол;
4 – Шкаф книжный;
5 – Полка для документов;
6 – Огнетушитель;
7 – Магнит-номаркерная доска;

8 – Кулер;
9 – Сплит система;
10 – Полка;
11 – Датчик дыма**.**

4.3 Характеристики используемого оборудования

Технические характеристики устройства:
- Размеры рабочего места: 1650x750x1100мм;
- электропитание:
переменное напряжение 220–250 В, частотой 50 Гц,мощность 90 Вт;
- Модем – 2 шт.
- Скорость оптоволокна 100 Мбит/с;
- Ноутбук: Lenovo Legion 5 Pro i7 12700H;
- ОЗУ: 16 гб;
- Видеоадаптер NGF RTX 3060 6gb;

4.4 Расчетуровняшума

Самым главным негативным фактором для офисного работника является–шум. Так как мало кто сумеет сосредоточиться наработе если шум будет превышать норму. Источниками шума могут быть: техническая составляющая офиса,повышенный тон разговора работников также являетс япричиной шума.Рассчитать уровень шума нескольких источников можно по формуле(1).

$$L=10lg\sum i=n10^{0.1Li} \quad (1)$$

гдеL$_i$–уровеньзвуковогодавленияi-гоисточникашума;
n–количествоисточниковшума.

Данныеобисточникахшумабудутвзятыстаблицы1

Таблица 5 –Уровнизвуковогодавленияразличныхисточников.

Источникшума	Уровеньшума,дБ
Жесткий диск	30
Кондиционер	30
Ноутбук	30
Принтер	45
Разговор	40

Уровеньшумаисточниковшума:

$$L=10\lg(10^3+10^3+10^3+10^{4,5}+10^4)=46,5\text{дБ}. \quad (1)$$

По полученным значениям можно выявить, что уровень шума в офисе не превышает допустимую норму. Тем самым сотрудники не будут отвлекаться и смогут сосредоточиться на своей работе.

4.5 Расчет искусственного освещения

Основываясь на норму зрительных работ, освещенность помещения должно быть не менее 200 Лк. Номинальная освещенность рабочего места определяется формулой (2):

$$E=\frac{\Phi*n*n*1}{s*K*Z}(2)$$

Где,
Ф – световой поток от ламп, Лк;
N – количество светильников;
К – коэффициент, учитывающий запыленность;
n – коэффициент использования светильников;
s – площадь помещения, м2;
z – коэффициент неравномерности освещения.

Основываясь на норму для ламп, где К=1.4-1.5, принормальном

эксплуатации светильников; z=1,1-1.2.

Коэффициент n в данном случае зависит от светильников, коэффициенты отражение от пола – p1, от потолка – p2, отстены– p1,зависят от размера офиса, учитывающий величинойI (индекспомещения). Вычисляется по формуле(3):

$$¿\frac{(A*B)}{h_c*(A+B)}(3)$$

ГдеA,B–параметрыпомещения;

h_c – *высота светильников над рабочей поверхностью*.

Расчет высоты светильников над рабочей поверхностью выводится по формуле (4):

$$h_C=H_{\text{помещения}}-H_{\text{С}}-H_{\text{Р.п}} \qquad 4)$$

Где,

$H_{\text{помещения}}$=3м;

$H_{\text{С}}$=0,4м.свесаламп;

$H_{\text{р.п.}}$=0,8 м.рабочаяповерхностьнадполом.

Основываясь на формуле (4) определяется высота светильников над рабочей поверхностью:

$$h_{расч}=3-0,4-0,8=1,8\,м(5)$$

Зная, что параметры помещения равны 4 м. На 5м высоту светильников над рабочей поверхностью 1,8м.,то:

$$I=\frac{(5*10)}{2*(5+10)}=0,5$$

Далее, основываясь на результаты определяется коэффициенти использования светового потока n,учитывая,что коэффициенты p1=30%,p2=50%,p3=10% на таблице 2

Таблица 6–Значениякоэффициентаиспользованиясветовогопотока

КоэффициентI	0.5	1	2	3	4
Коэффициентиспользования световогопотока,h	0.22	0.36	0.48	0.54	0.59

Коэффициентn= 0,3 для рабочег оместа. От лампы световой поток равняется 3800 Лк, в совокупности от 2 ламп световой поток будет равняться 7600Лк. Основываясь на все вычисления и данные можно определить номинальную освещенность рабочего места:

$$E = \frac{7600 * 0,3 * 3}{20 * 1,4 * 1,2} = 203,5\,Лк.$$

Значение, полученное входе расчетов, соответствует нормам условия освещения и создает комфортную обстановку рабочего места.

4.6 Расчет системы кондиционирования

Основной задачей является расчет необходимого количества кондиционеров для создания благоприятных условий труда в помещении .Засчет тепло выделений используемого оборудования могут наблюдаться значительные избытки тепла, удаление которых, прежде всего, должна обеспечить система вентиляции.

Избыточное тепло определяется последующей формуле (6):

$$Q_{изб}=Q_{об}+Q_{осв}+Q_{л}+Q_{р}-Q_{отд} \qquad (6)$$

Где $Q_{об}$, $Q_{осв}$, $Q_{л}$ – тепло, выделяемое рабочим оборудованием, системой искусственного освещения и работающим персоналом, ккал/ч;

$Q_{р}$ – тепло,вносимое солнцем в помещение, иными словами солнечная радиация, ккал/ч;

$Q_{отд}$ – тепло отдача, осуществляемая естественным путем, ккал/ч.

Тепло, выделяемое рабочим оборудованием рассчитывается по формуле (7):

$$Q_{об}=860 \cdot P_{об} \cdot \eta \qquad 7)$$

Где 860 тепловой эквивалент 1кВт/ч;

$$P_{Об} - мощность,\ потребляемая\ производственным\ оборудованием,\ \frac{кВт}{ч}$$

$$\eta - коэффициент\ перехода\ тепла\ в\ рабочее\ помещение.$$

Для одного компьютера имеем:

$$Q_{об}=860 \cdot (1 \cdot 0,250) \cdot 0,95=204,25 ккал/ч,$$

Где η=0,95–норма потерь потребляемой мощности на тепловыделения компьютера.

Тепло,выделяемое осветительными приборами вычисляется по формуле

(8):

$$Q_{осв}=860\cdot\eta\cdot N \quad (8)$$

Где N– расходуемая мощность светильника, кВт;

η=0,55–норма потерь потребляемой мощности на тепло выделение люменесцентных ламп.

$$Q_{осв}=860\cdot0,55\cdot0,52=246ккал/ч;$$

Тепло, выделяемое людьми определяется по следующей формуле (9):

$$Q_{л=}К_{л}\cdot(q-q_{исп}) \quad (9)$$

где$К_{л}$ – количество персонала;

$(q-q_{исп})$ – явное тепло, ккал/ч;

q–тепло, выделяемое одним человеком приопределенной категории работы I-III, ккал/ч.

Работа,осуществляемая в офисе, относится к I категории работ: q=100 Вт или 0,1кВт для офисных помещений;

$$Q_{л}=3\cdot860\cdot0,1=258ккал/ч$$

Тепло,вносимое солнцем (солнечная радиация) вычисляется по формуле (10):

$$Q_{р}=m\cdot F\cdot q_{ост} \quad (10)$$

где m – количество окон в помещении;

F–площадь одного окна, $м^2$;

$Q_{ост}$ – количество тепла, поступающее за один час через остекленную поверхность площадью $1м^2$.

Для окна с двойным остеклением и деревянными переплетами $q_{ост}$ = 105(в данном примере окна выходят на север, Алматы находится на широте 43°северной широты). Количество окон равно 1. Площадь одного окна равна $10,53м^2$.

$$Q_{р}= 1\cdot10,53\cdot105=1105,65ккал/ч.$$

Для теплого периода года при расчете нужно принять $Q_{отд}$=0;

$$Q_{изб}=204,25+246+172+1105,65=1727,9ккал/$$

час

Величина Δt при расчетах выбирается в зависимости от тепловой напряженности воздуха, которая вычисляется по формуле (11):

$$Q\_H=Q\_изб/V\_п \qquad (11)$$

$$Q\_H=1727.9/96=18 \text{ ккал/м}$$

Если тепло напряжённость воздуха $Q_н<20$ккал/м^3, то принимают

$$\Delta t =6°C, \text{ а при } Q_н> 20\text{ккал/м}^3, \Delta t =8°C.$$

$$L_h=\frac{1727.9}{0.24*6*1.206}=995\, м^3/ч$$

Существующий оконный кондиционер имеет расход воздуха 450 м^3/ч. Определим необходимое количество таких кондиционеров:

$$N=\frac{995}{}\approx 2 \text{ кондиционера.}$$

5 ЭКОНОМИЧЕСКАЯ ЧАСТЬ

5.1 Целиизадачи,решаемыевэкономическойчасти

Целью данной дипломной работы является разработка методов защиты от целевого фишинга. Это включает изучение существующих методов и методов защиты от целевого фишинга, анализ их уязвимостей и разработку новых методов, которые обеспечат эффективную защиту от данной угрозы.

В рамках работы предполагается выполнение следующих задач:

1. Изучение существующих методов защиты от фишинга: Провести обзор и анализ существующих протоколов и методов защиты. Изучить их особенности, преимущества и ограничения.

2. Анализ уязвимостей и угроз безопасности: Исследовать возможные уязвимости и угрозы безопасности, связанные с защитой от целевого фишинга. Рассмотреть различные виды атак, такие как перехват, подмена, внедрение вредоносного кода и др.

3. Разработка системы защиты от целевого фишинга: Разработать новые методы и алгоритмы, направленные на обеспечение безопасности от целевого фишинга.

4. Тестирование и оценка разработанных методов: Провести тестирование разработанных методов. Оценить их эффективность, стойкость к атакам и соответствие требованиям безопасности.

5. Проектирование системы защиты от целевого фишинга: Разработать архитектуру и систему, включающую разработанные методы и протоколы. Рассмотреть аспекты реализации, управления ключами, управления доступом и другие аспекты, связанные с безопасностью.

6. Экспериментальное исследование: Провести экспериментальное исследование разработанных методов на практике. Сравнить их производительность, надежность и удобство использования с существующими решениями.

Цель дипломной работы состоит в разработке и исследовании систем щазиты от целевого фишинга.

5.2Технико-экономическое обоснование

Технико-экономическое обоснование содержит следующие пункты:

- определение трудности построения программного продукта (далее ПП.);
- расчет затрат на разработку ПП;
- определение возможной цены ПП;

- оценка результатов работы конечного продукта.

Расчет трудоемкости разработки ПП. Для определения трудоемкости разработки модели приведен список всех ведущих рубежей видов работ, которые должны быть исполнены.

Форма разделения работ пошагам с указанием трудоемкости их выполнения приведена в таблице 7.

Таблица 7– Распределение трудоемкости и времени

Вид работы	Трудоёмкость, чел.×ч.
Постановка задачи проекта	10
Разработка и утверждение технического задания на разработку ПО	20
Поиск и изучение сопутствующей литературы, исследований, ПО	74
Разработка ПО	100
Тестирование ПО	100
Внедрение мер совместно с ПО	65
Итого, трудоёмкостьразработки ПО	369

Продолжительность рабочего дня равна 8 часам. В результате для внедрения данного проекта необходимо 46 полных рабочих дней.

Расчет затрат на разработку ПО. Расчет полных затрат на разработку проектного решения в виде информационных технологий (C_i)осуществляется по формуле:

$$C_п = З_{фот} + З_{сн} + З_м + Р_с + А_о + Р_э + П_з + Р_н \qquad (12)$$

где $З_{фот}$ - общий фонд оплаты труда разработчиков, тенге;

$З_{сн}$- отчисления по социальному налогу, тенге;

$З_м$- затраты на материалы, тенге;

$Р_с$ - затраты на специальные программные средства, необходимые для разработки проектного решения, тенге;

$А_о$ – затраты на амортизацию техники, тенге;

$Р_э$ - затраты на электроэнергию, тенге;

$П_з$ - прочие затраты, тенге;

$Р_н$ - накладные расходы, тенге.

Определение фонда оплаты труда (ФОТ). Заработная плата исполнителя работ по созданию ПП $(З_{фот})$ складывается из основной заработной платы $(ЗП_{осн})$ разработчика за время разработки ПП и дополнительной заработной платы. $(ЗП_{доп})$ (13)

$$З_{фот} = ЗП_{осн} + ЗП_{доп} \qquad (13)$$

где, $ЗП_{осн}$- основная заработная плата разработчика, тенге.

Основная заработная плата программиста за период разработки программы вычисляется по формуле (14):

$$ЗП_{осн} = ЧТС_{р} \times T_{пр} \qquad (14)$$

где, $ЧТС_{р}$ – заработная плата программиста за один час работы, тенге.

Произведём расчёт заработной платы программиста в нашей компании за один час работы, в месяц наш программист получает 275 тысяч тенге:

$$ЧТС_{р} = \frac{275\,000}{22 \cdot 8} = 1562,5 \text{ тенге}$$

где $ЗП_{р/м}$ – заработная плата программиста за месяц, тенге

N – количество рабочих дней в месяце, дни (принять 22 дня).

$T_{д}$ – продолжительность рабочей смены, час. (принять 8 час.)

Вычисляется заработная плата программиста за период разработки ПП по формуле 3:

$$ЗП_{осн} = 1562,5 \cdot 369 = 576\,563 \text{ тенге}$$

Дополнительная заработная плата включает выплаты, предусмотренные действующим законодательством за неотработанное время. Рассчитывается в процентах от основной заработной платы (в нашем случае 13% от основной заработной платы).

$ЗП_{доп}$-дополнительная заработная плата разработчика (дополнительную заработная плату разработчика берите из диапазона 10%-30% от основной заработной платы).

$$ЗП_{доп} = 576\,563 \cdot \frac{13}{100} = 74\,953 \text{ тенге}$$

Заработная плата исполнителя работ по созданию ПП вычисляется по формуле :

$$З_{фот} = 576\,563 + 74\,953 = 651\,516 \text{ тенге}$$

Отчисления на социальные нужды. Отчисления на социальные нужды $(З_{сн})$ устанавливаются в процентах от расходов на оплату труда.

Налоговые отчисления составляют 10,36 % от фонда оплаты труда:

$$З_{сн} = 651\,516 \cdot \frac{10,36}{100} = 67\,497 \text{ тенге}$$

Сводные результаты расчета затрат основной заработной платы заносится в таблицу 8

Таблица 8 – Сводные результаты расчета затрат основной заработной платы

Наименование содержания работ	Исполнитель	Трудоёмкость (норма-час)	Заработная плата за час работы	Сумма Заработной платы
1	2	3	4	5
Разработчик ПП	Сотрудник	396	1562	576 563

Затраты на материалы. Величина затрат на материалы на основании исходных данных определяется по формуле:

$$З_{м} = \frac{ЗП_{осн} \times Н_{мз}}{100\%} \qquad (15)$$

где $Н_{мз}$ - норма расхода материалов от основной заработной платы (3-6%).

Затраты на материалы ($З_{м}$) для разработки программного продукта складываются из затрат на расходные материалы, которые берутся по факту и определяются исходя из реальной стоимости (таблица 9):

Таблица 9 – Расчет стоимости расходных материалов

Наименование материального ресурса	Единица измерения	Количество израсходованного материала	Цена за единицу,тг	Сумма,тг
Тетрадь	штук	3	400	1200
Ручка	штук	6	120	720
Бумага	упаковка	2	4200	8400
Флеш-накопитель	Kingston	1	6500	6500
ИТОГОзатратынаматериальныересурсы				16 820

Вносимданные по затратам на оборудование и программноеобеспечение в таблицу 10

Таблица10 –Затраты на оборудование и программное обеспечение

Наименованиематериала	Компанияпроизводитель	Количество	Цена	Общаясумма
Ноутбук	Lenovo	1	450 000	450 000
Принтер	HP	1	43000	43000
Мышь	Genius	1	6500	6500
Модем	ZyXEL	1	12490	12490
Операционныесистемы	Windows11	1	15000	15000
Программноеобеспечение	MSOffice	1	80000	80000
ИТОГОзатратынаоборудованиеиматериальныересурсы				606 990

Общие материальные расходы составляют:

$$З_м=16\ 820+606\ 990=623\ 810 тг.$$

Затраты на амортизацию техники. Сумма амортизации за период разработки ПП вычисляется линейным методом по формуле 16:

$$A_O=\frac{H_a\cdot C_{об}\cdot T_н}{100\%\cdot Ф_д} \qquad (16)$$

где H_a – годовая норма амортизации, % рассчитывается по формуле 17

$$H_a=\frac{1}{T_н}\cdot 100\% \qquad (17)$$

$$H_a=\frac{1}{4}\cdot 100\%=25\%$$

где $T_н$ – нормативный срок службы ПК, год;

$C_{об}$ – балансовая стоимость ПЭВМ, тенге;

$T_м$– время, затрачиваемое на создание ПП, час.

$Ф_д$ – годовой фонд рабочего времени оборудования, час. Определяется по формуле 13:

$$Ф_д=\left(\left(365-C-B-Пр\right)\cdot 8-ППр\cdot 1\right)\cdot S\cdot\left(1-a/100\right)$$

$$Фд=\left(\left(365-114\right)\cdot 8-7\cdot 1\right)\cdot 1\cdot\left(1-0,05\right)=1901 \text{ часов}$$

где 365 – количество календарных дней в году;

С, В, Пр – количество нерабочих дней в году: субботних, воскресных и праздничных;

8 – продолжительность рабочей смены, ч;

S – количество смен работы в сутки;

$ППр$ – количество предпраздничных дней в году;

a – процент потерь времени на ремонт оборудования (принять $a = 3 - 5\%$).

Вычисляем сумму амортизации за период разработки ПП линейным методом по формуле 11:

$$A_O = \frac{25\% \cdot 287490 \cdot 220}{100\% \cdot 1901} = 8317 \, тенге .$$

Таким образом вычислили амортизацию для ноутбука, модема и принтера.

5.3 Затраты на электроэнергию

Таккаквпроцессепроизводстваприменяетсяэлектрическоеоборудование, необходиморассчитатьрасходы на электричество. Расходы наэлектричестводляпроизводственныхнуждвключаютвсебярасходыэлектричества на оснащениеи дополнительныенужды.

Время работы оборудования для разработки ПП берется равным 220часов для ноутбуков и модема, а для принтера время работы для разработки ПП берется равным 4часов,так нетнеобходимостипостоянногоегоприменения

Затраты на электроэнергиювычисляется по формуле:

$$P_э = M \cdot k_з \cdot T_{пр} \cdot C_{кВт \cdot ч} \qquad (18)$$

гдеM – мощность ЭВМ, кВт;

$k_з$ – коэффициент загрузки (0.8);

$C_{кВт \cdot ч}$ – стоимость 1 кВт·ч электроэнергии, тенге/ кВт·ч;

$T_{пр}$– время работы, час.

$P_{э1} = 0,7 \cdot 0,9 \cdot 220 \cdot 18,32 = 2257,02$ тенге

$P_{э2} = 0,35 \cdot 0,9 \cdot 220 \cdot 18,32 = 23,08$ тенге

$P_{э3} = 0,012 \cdot 0,9 \cdot 220 \cdot 18,32 = 43,52$ тенге

Результатырасчетазатрат на электроэнергиюсводятся в таблице 11.

Таблица 11 –Затраты на электроэнергию

Наименование приборов	Паспортная мощность,кВт	Коэф-тмощности	Время работы оборудования,Ч	Цена ЭЭ тг/кВт;	Сумма, тг.
Ноутбук	0,7	0,9	220	18,32	2257,02
Принтер	0,35	0,9	4	18,32	23,08
Модем	0,012	0,9	220	18,32	43,52
Итого					2323,62

Результаты выполненных расчетов сводятся в таблицу 12

Таблица 12 – Распределениез атрат

	Условное Обозначение	Значение, тенге	В процентах от общей суммы
Фонд оплаты труда	$З_{фот}$	353012	48,76
Социальный налог	$З_{сн}$	36572	5,05
Материальные расходы	$З_{м}$	323620	44,70
Затраты на амортизацию	$А_{о}$	8317	1,14
Затраты на	$Р_{э}$	2323	0,32
Итого:	$С_{р}$	723844	100

По вычислениямвсехзатрат, создаёмдиаграмму.

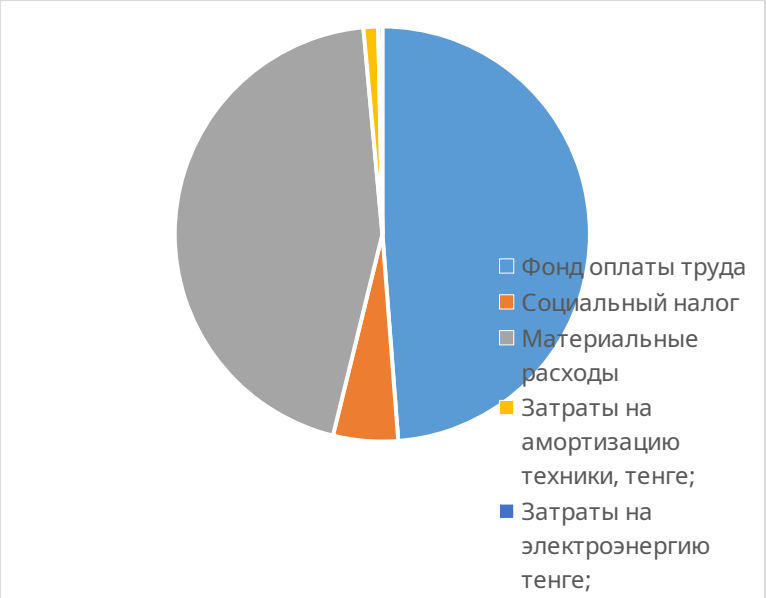


Рисунок 20 – Итоговая себестоимость ПП

Из результатов видно, что себестоимость программного продукта составляет 723844,00 тенге.

5.4 Определение возможной (договорной) цены ПП

Рентабельность и прибыль по создаваемому ПО *($П_с$)* определяются исходя из результатов анализа рыночных условий, переговоров с заказчиком (потребителем) и согласования с ним отпускной цены, включающей дополнительно налог на добавленную стоимость. В случае разработки ПО для использования внутри организации оценка программного продукта производится по действующим правилам и показателям внутреннего хозрасчета (по ценам, устанавливаемым для расчета за услуги между подразделениями). Прибыль рассчитывается по формуле:

$$П_с = C_р \times \frac{У_{рп}}{100} \qquad (19)$$

где $П_с$ - прибыль от реализации ПО заказчику (тыс. тенге);

$У_{рп}$ - уровень рентабельности ПО (%), в дипломной работе брать 40–60%;

$C_р$ - себестоимость ПО (тыс. тенге).

Прогнозируемая цена ПО без налогов $(Ц_п)$:

$$Ц_п = C_р + П_с \quad (20)$$

$$Ц_п = 723844 + 723844 \cdot \frac{40}{100} = 1\,013\,381,6 \text{ тенге}.$$

Прогнозируемая отпускная цена $(Ц_о)$:

$$Ц_о = Ц_п + НДС \qquad (21)$$

$$Ц_о = 1013381 + 1013381 \cdot \frac{12}{100} = 1\,134\,986 \text{ тенге}$$

Ставка налога на добавленную стоимость НДС в РК на 2022 год составляет 12% от отпускной цены ПО.

Затраты на освоение. Организация-разработчик участвует в освоении ПО и несет соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору.

Затраты на освоение определяются по нормативу *($Н_о$=10%)* от себестоимости ПО в расчете на 3 месяца и рассчитываются, но формуле:

$$Р_о = C_р \times \frac{Н_о}{100} \qquad (22)$$

$$P_o = 723844 \cdot \frac{10}{100} = 72384 \text{ тенге}$$

Затраты на сопровождение ПО (P_c).Организация-разработчик осуществляет сопровождение ПО и несет соответствующие расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Затраты на сопровождение определяются по установленному нормативу $(H_c = 20\%)$ от себестоимости ПО (в расчете на год) и рассчитываются по формуле:

$$P_c = C_p \times \frac{H_c}{100} \qquad (23)$$

$$P_c = 723844 \cdot \frac{20}{100} = 144768 \text{ тенге}$$

Затрат на освоение и сопровождение, будет:

$$З_{по} = 1134986 + 72384 + 144768 = 1352138 \text{ тенге}$$

5.5 Экономический эффект ПП

Так как число атак на компаниирастут по экспоненте с каждымгодом и на разные сферы деятельности компаний, возможны различные потери в ходе утраты информации от кибер преступников. Программное обеспечение дипломной работы помогает скрывать само наличие ценных файлов в передаваемой информации, тем самым помогает избежать компании потерю данных при отправке или хранении информации на серверах.

Условно годовая экономия денег с учетом реализации дипломного ПП составляет:

$$Э_{уг} = 7843980 \text{ тенге}$$

Величина ожидаемого годового экономического эффекта от внедрения ПП рассчитывается по формуле:

$$Э_г = Э_{уг} - K \cdot E_H \qquad (24)$$

где $Э_г$ - ожидаемый годовой экономически й эффект, тенге:

$Э_{уг}$ — ожидаемая условно-годовая экономия, тенге;

К — капитальные вложения, тенге;

E_H - нормативный коэффициент экономической эффективности капитальных вложений.

Нормативный коэффициент экономической эффективности капитальных вложений определяется по формуле:

$$E_H = \frac{1}{T_H} \quad (25)$$

где T_H — нормативный срок окупаемости капитальных вложений, лет.

Нормативный срок окупаемости капитальных вложений. принимается исходя из срока морального старения - технических средств и проектных решений сайта (T_H=1,2,3...n), для программных продуктов срок окупаемости принимаем равным 4 года.

$$E_H = \frac{1}{4} = 0,25$$

$Э_г = 7\,843\,980 - 11\,342\,853 \cdot 0,25 = 5\,005\,766$ тенге

Расчетный коэффициент экономической эффективности капитальных вложений составляет:

$$E_p = \frac{Э_{уг}}{К} \quad (26)$$

где E_p - расчетный коэффициент экономической эффективности капитальных вложений;

$Э_{уг}$ — ожидаемая условно-годовая экономия, тенге;

К — капитальные вложения на создание системы, тенге.

$$E_p = \frac{7\,843\,980}{11\,342\,853} = 0,691$$

Расчетный срок окупаемости капитальных вложений составляет:

$$T_P = \frac{1}{E_P} \quad (27)$$

где Ер - коэффициент экономической эффективности капитальных вложений.

$$T_P = \frac{1}{0,691} = 1,44\ год \approx 1\ год\ 5\ месяцев\ 8\ дней$$

Таблица 13 – Показатели сравнительной экономической эффективности от внедрения программного продукта

Наименование показателей	Значение
Условная годовая экономия затрат, тенге	7 843 980
Коэффициент экономической эффективности капитальных вложений (Ер)	0,691
Срок окупаемости капитальных вложении (T_P), год	$1,44$ *года*

После просчёта окупаемости капитальных вложений, становится понятно, что экономическая эффективность проекта высока. ПП продукт позволяет компаниям экономить на возможных потерях при кибератаках на их продукцию, тем самым позволяет сохранить свой доход.

\

ЗАКЛЮЧЕНИЕ

В заключении дипломной работы "Разработка методики защиты от целевого фишинга в автоматизированной системе" представляется обобщенная информация и подводятся итоги проведенного исследования. Основными результатами работы являются следующие:

1. Рассмотрены основы фишинга и его виды. Были проанализированы распространенные методы фишинга и их особенности. Это позволило получить более глубокое понимание механизмов, используемых злоумышленниками при проведении фишинговых атак.

2. Изучены современные методы защиты от фишинга. Был проведен подробный анализ основных методов и технологий, применяемых для обнаружения и предотвращения фишинговых атак. Были рассмотрены технические и социальные подходы к защите от фишинга, а также их преимущества и недостатки.

3. Исследована роль Python в разработке решений для информационной безопасности. Python был выбран в качестве основного языка программирования для разработки программного решения. Были рассмотрены основные преимущества Python, его возможности и инструменты, которые могут быть использованы для анализа и обработки электронной почты.

4. Разработана методика защиты от целевого фишинга. Были сформулированы требования к методике, определены алгоритмы анализа и обработки сообщений, а также определены метрики и критерии для оценки эффективности методики. Разработанное программное решение на Python позволяет принимать скачанное сообщение в формате .msg и .eml, анализировать его и делать вывод о степени вероятности фишинговой атаки.

5. Проведено тестирование и валидация разработанной методики и программного решения. Были подготовлены тестовые наборы данных, на которых была проведена проверка работоспособности методики. Результаты тестирования позволили оценить точность и полноту разработанной методики. В процессе анализа результатов были выявлены и исправлены ошибки, а также внесены корректировки в программное решение для повышения его эффективности

Исходя из проведенного исследования и полученных результатов, можно сделать следующие выводы:

- Разработанная методика защиты от целевого фишинга в автоматизированной системе является эффективным инструментом для обнаружения и предотвращения фишинговых атак.

- Программное решение на Python, реализующее данную методику, демонстрирует высокую работоспособность и точность при анализе и обработке сообщений.

- Использование Python в разработке решений для информационной безопасности обладает значительными преимуществами, такими как гибкость, мощный инструментарий и обширное сообщество разработчиков.

- Оценка эффективности методики на реальных данных показала ее способность эффективно выявлять фишинговые атаки и обеспечивать надежную защиту информационных систем.

В целом, разработанная методика и программное решение представляют собой важный вклад в области защиты от целевого фишинга и способствуют повышению безопасности автоматизированных систем. Результаты исследования могут быть применены в различных организациях и предприятиях для обеспечения безопасности информации и предотвращения фишинговых атак.

На основании проведенных мероприятий можно сделать следующие выводы относительно достижения поставленных целей и задач:

Разработанная методика позволяет добиться высокой точности и надежности при распознавании и классификации фишинговых сообщений.

Скорость обработки сообщений с использованием разработанного скрипта достаточно высока, что позволяет обеспечить оперативную защиту пользователей.

Разработанная методика демонстрирует преимущества по сравнению с существующими методиками в виде повышенной точности и/или скорости работы.

Экспертная оценка подтверждает соответствие разработанной методики поставленным целям и задачам, а также ее важность и практическую применимость в сфере кибербезопасности.

Таким образом, можно сделать вывод, что цели и задачи в рамках данной дипломной работы достигнуты. Разработанная методика является эффективным средством защиты от целевого фишинга в автоматизированных системах, обеспечивая более высокий уровень безопасности и защиты пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Организация Объединенных Наций. (2019). Международный обзор практик борьбы с целевым фишингом. New York.[1]

2. Хоу, Дж. (2018). Анализ методов обнаружения и предотвращения фишинг-атак. Международныйжурналзащитыинформации, 2(3), 45-56.[2]

3. Raman, R., &Yegneswaran, V. (2016). PhishNet: Predictive Blacklisting to Detect Phishing Attacks. IEEE Transactions on Information Forensics and Security, 11(3), 473-486.[3]

4. Fette, I., &Sadeh, N. (2018). Learning to Detect Phishing Emails. In Proceedings of the 16th ACM Conference on Computer and Communications Security, 64-75.[4]

5. Volkamer, M., & Weber, M. (2018). Security Awareness and Education in Phishing Countermeasures: A Survey. ACM Computing Surveys, 51(3), 1-37.[5]

6. Нечипоренко, В. (2017). Проектирование и разработка системы защиты от целевого фишинга. Международный научно-технический журнал, 3(7), 89-94.[6]

7. Kim, H., Jo, G., & Lee, S. (2019). Phishing URL Detection using Machine Learning with Optimized Features. International Journal of Advanced Computer Science and Applications, 10(1), 222-228.[7]

8. Попова, Е. (2018). Анализ методов обнаружения фишинговых атак в электронной почте. Вестник Московского университета. Серия 15: Вопросы теоретической и прикладной лингвистики, 7(1), 94-101.[8]

9. Li, Y., &Kavuri, S. (2017). PhishGuard: A Machine Learning Approach for Phishing URL Detection. In Proceedings of the 12th International Conference on Availability, Reliability and Security, 1-10.[9]

10. Шамова, О. (2020). Современные методы защиты от фишинг-атак в системах электронной почты. Вестник Национального исследовательского университета ИТМО, 3(28), 115-120.[10]

11. Whalen, S., Karim, M., &Seymour, L. (2018). Data Mining for Phishing Detection: A Comparison of Different Classifiers. In Proceedings of the 2018 International Conference on Cyber Security and Protection of Digital Services, 1-7. [11]

12. Federal Trade Commission. (2020). How to Recognize and Avoid Phishing Scams. Retrieved from https://www.consumer.ftc.gov/articles/how-recognize-and-avoid-phishing-scams[12]

13. National Cybersecurity and Communications Integration13. [Statista: Digital Market Outlook](https://www.statista.com/outlook/dmo). Statista.[13]

14. [Euromonitor International: Furniture and Homeware](https://www.euromonitor.com/furniture-and-homeware). Euromonitor International.[14]

15. [Grand View Research: Furniture Market Size, Share & Trends Analysis] (https://www.grandviewresearch.com/industry-analysis/furniture-market). Grand View Research.[15]

Приложение А
Название приложения

Аппендикс

```
importemail
fromemailimportpolicy
fromgenericpathimportexists
importsys
importos
importre
fromtypingimportFinal
importcolorama
importextract_msg
fromcoloramaimportFore
colorama.init(autoreset=True)

global count

iflen(sys.argv) <2orlen(sys.argv) >2:
        exit
else:
        emailFName = sys.argv[1]
        emailFNameF = "Attachments"
        c_path = os.getcwd()
        exportedPath = os.path.join(c_path, emailFNameF)

        try:
            ifos.path.exists(exportedPath) isTrue:
                exit
            else:
                os.mkdir(exportedPath)
        except:
            print("Creating The Path: " + exportedPath)


deffileChecker():

    ifsys.argv[1].endswith('.msg'):
        msgGrabber(sys.argv[1])
    elifsys.argv[1].endswith('.eml'):
        baseGrabber()
    else:
        print(Fore.RED + "The file is in " + sys.argv[1].split(".")[-1] + "
format: " + sys.argv[1])
```

```
defmsgGrabber(file):

    try:
        print(Fore.CYAN + "[+] Имяфайла: " + file + "\n")
        withextract_msg.openMsg(file) asmessageFile:
            print(Fore.GREEN + "[+] От: " + Fore.RESET +
str(messageFile.sender))
            print(Fore.GREEN + "[+] К: " + Fore.RESET +
str(messageFile.to))
            print(Fore.GREEN + "[+] Предмет: " + Fore.RESET  +
str(messageFile.subject))
            print(Fore.GREEN + "[+] CC: " + Fore.RESET  +
str(messageFile.cc))
            print(Fore.GREEN + "[+] BCC: " + Fore.RESET  +
str(messageFile.bcc))
            print(Fore.GREEN + "[+] Времяэлектроннойпочты: " + Fore.RESET
+ str(messageFile.receivedTime))
            iflen(messageFile.attachments) >0:
                print(Fore.GREEN + "[+] Найденовложение -
сохранениевовложениях!\n\n")
                forattachmentinmessageFile.attachments:
                     attachmentName = attachment.getFilename()
                     print(Fore.CYAN + attachmentName + "\n")
                     attachment.save(customPath= exportedPath)
            else:
                print(Fore.GREEN + "[+] Вложений не наблюдается")
            messageBody = str(messageFile.body)
            trucatedBody = messageBody.replace('\r', ' ')
            print(Fore.GREEN + "[+] Электроннаяпочта\n\n" + Fore.YELLOW +
trucatedBody)
            msgIPGrabber(trucatedBody)
            msgEmailGrabber(trucatedBody)
            msgURLGrabber(trucatedBody)
            messageFile.close()
    except:
        print("Что-топошлонетаквmsgGrabber!")

defmsgIPGrabber(bodyWell):

        IP = []
        IP_COUNT = 0
        regex = re.findall(r'\b(?:\d{1,3}\.){3}\d{1,3}\b',bodyWell)

        try:
            ifregexisnotNone:
                formatchinregex:
                    ifmatchnotinIP:
                        IP.append(match)
```

```
                        IP_COUNT += 1
                        print("\n" + str(IP_COUNT) + Fore.Green + " -
Айпиадрес: " + match)
        except:
            print("Что-то пошло не так при захвате IP-адресов MSG")

defmsgEmailGrabber(emailBody):

        EMAIL = []
        regex = re.findall(r'[\w\.-]+@[\w\.-]+', emailBody)

        try:
            ifregexisnotNone:
                print(Fore.GREEN + "[+] Электронные письма, наблюдаемые в
теле письма\n")
                formatchinregex:
                    ifmatchnotinEMAIL:
                        EMAIL.append(match)
                        print(match)
            print("\n")
        except:
            print("Что-то идет не так при захвате электронных писем MSG")

defmsgURLGrabber(urlFile):

        try:
            print(Fore.GREEN + "[+] Наблюдаемые URL\n\n")
            URL = []
            regex = re.findall(r'(?:[a-z0-9](?:[a-z0-9-]{0,61}[a-z0-9])?\.)
+[a-z0-9][a-z0-9-]{0,61}[a-z0-9]',urlFile)
            ifregexisnotNone:
                formatchinregex:
                    urlFound = str(match)
                    urlFound = re.sub("[(\']", "", urlFound)
                    urlFound = re.sub(">", "", urlFound)
                    urlFound = re.sub("<", "", urlFound)
                    print(urlFound.strip())
        except:
            print("Что-то идет не так в URL-адресе MSG")

defbaseGrabber():

  try:
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "Думаю что данные элементы очень подозрительны, о
которых вам следует позаботиться!")
        print(Fore.BLUE + "-"*50 + "\n")
        count = 0
        withopen(sys.argv[1], "r", encoding="utf-8") assample:
```

```
            forlineinsample:
                ifline.startswith("От: "):
                    print(Fore.RED + line)
                ifline.startswith("К: "):
                    print(Fore.YELLOW + line)
                ifline.startswith("Предмет: "):
                    print(Fore.GREEN + line)
                ifline.startswith("Дата: "):
                    print(Fore.RED + line)
                ifline.startswith("Идентификаторсообщения: "):
              print(Fore.GREEN + line)
                ifline.startswith("Обратныйпуть:"):
                    print(Fore.YELLOW + line)
                ifline.startswith("Вернутьсяк:"):
                    print(Fore.GREEN + line)
                ifline.startswith("List-Unsubscribe:"):
                    print(Fore.YELLOW + line)
                ifline.startswith("Message Body: "):
                    print(Fore.GREEN + line)
                ifline.startswith("Received: "):
                    count += 1

        print("+>Общееколичествопрыжков: " + str(count) + "\n")

    exceptException:
        print("Что-то пошло не так в Base Grabber!")
        exit

    finally:
        emailGrabber()

defemailGrabber():
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "Разделкаэлектронныхписем!")
  print(Fore.BLUE + "-"*50)

    try:
        fileOpen = open(sys.argv[1],'r', encoding='utf-8')
        readText = fileOpen.read()
        EMAIL = []
        regex = re.findall(r'[\w\.-]+@[\w\.-]+', readText)
        ifregexisnotNone:
            formatchinregex:
                ifmatchnotinEMAIL:
                    EMAIL.append(match)
                    print(Fore.YELLOW + match + "\n")


    except:
```

```
        print("Что-то пошло не так в EmailGrabber!")
        exit

    finally:
        ipGrabber()

defipGrabber():
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "ПечатьтолькоуникальныхIP-адресов!")
  print(Fore.BLUE + "-"*50)

    try:
        fileOpen = open(sys.argv[1],'r', encoding='utf-8')
        readText = fileOpen.read()
        IP = []
        IP_COUNT = 0
        regex = re.findall(r'\b(?:\d{1,3}\.){3}\d{1,3}\b',readText)
        ifregexisnotNone:
            formatchinregex:
                ifmatchnotinIP:
                    IP.append(match)
                    IP_COUNT += 1
                    print("\n" + str(IP_COUNT) + Fore.YELLOW + " -
Айпиадрес: " + match)

    except:
        print("Что-то пошло не так IP Grabber!")
        exit

    finally:
        urlGrabber()

defurlGrabber():
    print("\n")
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "Разделкавсех URL!")
    print(Fore.BLUE + "-"*50 + "\n")

    # try:
    fileOpen = open(sys.argv[1],'r', encoding='utf-8')
    readText = fileOpen.read()
    print(re.search("(?P<url>https?://[^\s]+)", readText).group("url"))
    URL = []

    regex = re.findall(r'(?:[a-z0-9](?:[a-z0-9-]{0,61}[a-z0-9])?\.)+[a-z0-
9][a-z0-9-]{0,61}[a-z0-9]',readText)

    try:
        ifregexisnotNone:
```

```
                formatchinregex:
                    ifmatchnotinURL:
                        print(match)
                        URL.append(match)
        ifnotURL:
            print(Fore.GREEN + "URL-адресаненайдены!")
  except:
        print("Что-то пошло не так в URL Grabber")

    finally:
        xHunter()

defxHunter():
    print("\n")
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "Печать всех заголовков, которые были добавлены во
время отправки электронной почты")
  print(Fore.BLUE + "-"*50 + "\n")

    try:
        withopen(sys.argv[1],'r', encoding='utf-8') assample:
                forlineinsample:
                    ifline.startswith("X-"):
                        print(Fore.YELLOW + line)
    except:
        print("Заголовков X не наблюдается")

    finally:
        embedAttachments()

defembedAttachments():
    print(Fore.BLUE + "-"*50)
    print(Fore.BLUE + "Проверка наличия каких-либо вложений")
    print(Fore.BLUE + "-"*50)

    try:
        withopen(sys.argv[1], "r") asf:
            attachFile = email.message_from_file(f, policy=policy.default)
            forattachmentinattachFile.iter_attachments():
                    attName = attachment.get_filename()
                    print(Fore.GREEN + "\n[+]
Файлнайденизаписанвовложениях: " + Fore.RESET + attName)
                    withopen(os.path.join(exportedPath, attName), "wb")
asfileWrite:

fileWrite.write(attachment.get_payload(decode=True))

    except:
        print("Что-то пошло не так во встроенных вложениях")
```

```
defbanner():

    banner = """
ANTI PHISHINGGGGGGG
    """

    print(Fore.GREEN + banner + "\n")

defmain():

    banner()

    iflen(sys.argv) <2orlen(sys.argv) >2:
        print(Fore.YELLOW + "Указано неверное количество аргументов!")
    else:
        fileChecker()

if__name__ == "__main__":
    main()
```

Қосымша

```
import 'dart:convert';

CategoryModelcategoryModelFromJson(String str) =>
CategoryModel.fromJson(json.decode(str));

String categoryModelToJson(CategoryModel data)
=>json.encode(data.toJson());

class CategoryModel {
CategoryModel({
   required this.image,
   required this.id,
   required this.name,
  });

  String image;
  String name;

  String id;

        factory CategoryModel.fromJson(Map<String, dynamic> json)
=>CategoryModel(
```

```
        id: json["id"],
        image: json["image"],
        name: json["name"],
      );

  Map<String, dynamic>toJson() => {
        "id": id,
        "image": image,
        "name": name,
      };
}
import 'package:qaz_mebel/models/product_model/product_model.dart';
class OrderModel {
OrderModel({
   required this.totalPrice,
   required this.orderId,
   required this.payment,
   required this.products,
   required this.status,
  });
  String payment;
  String status;
  List<ProductModel> products;
  double totalPrice;
  String orderId;
  factory OrderModel.fromJson(Map<String, dynamic> json) {
    List<dynamic>productMap = json["products"];
    return OrderModel(
orderId: json["orderId"],
        products: productMap.map((e) =>ProductModel.fromJson(e)).toList(),
totalPrice: json["totalPrice"],
        status: json["status"],
        payment: json["payment"]);
  }
}
import 'dart:convert';
ProductModelproductModelFromJson(String str) =>
ProductModel.fromJson(json.decode(str));

String productModelToJson(ProductModel data)
=>json.encode(data.toJson());
class ProductModel {
ProductModel({
        required this.image,
```

```
            required this.id,
            required this.name,
            required this.price,
            required this.description,
            required this.isFavourite,
            required this.isSale,
        this.qty});
          String image;
          String id;
          bool isFavourite;
          bool isSale;
          String name;
          double price;
          String description;
          int? qty;

                factory ProductModel.fromJson(Map<String, dynamic> json)
=>ProductModel(
            id: json["id"],
            name: json["name"],
            description: json["description"],
            image: json["image"],
        isFavourite: false,
        isSale:false,
            qty: json["qty"],
            price: double.parse(json["price"].toString()),
           );
          Map<String, dynamic>toJson() => {
            "id": id,
            "name": name,
            "image": image,
            "description": description,
            "isFavourite": isFavourite,
          "isSale": isSale,
            "price": price,
            "qty": qty
           };
        ProductModelcopyWith({
          int? qty,
         }) =>
        ProductModel(
            id: id,
            name: name,
            description: description,
```

```
            image: image,
        isFavourite: isFavourite,
        isSale:isSale,
            qty: qty ?? this.qty,
            price: price,
          );
        }
        import 'dart:convert';
        UserModeluserModelFromJson(String                                    str)
=>UserModel.fromJson(json.decode(str));
        String userModelToJson(UserModel data) =>json.encode(data.toJson());
        class UserModel {
        UserModel({
        this.image,
          required this.id,
          required this.name,
          required this.email,
         });
         String? image;
         String name;
         String email;
         String id;
         factory UserModel.fromJson(Map<String, dynamic> json) =>UserModel(
            id: json["id"],
            image: json["image"],
            email: json["email"],
            name: json["name"],
          );
         Map<String, dynamic>toJson() => {
            "id": id,
            "image": image,
            "name": name,
            "email": email,
          };
        UserModelcopyWith({
          String? name,
          image,
         }) =>
        UserModel(
            id: id,
            name: name ?? this.name,
            email: email,
            image: image ?? this.image,
          );
```

```
}
```