ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»
ПО ПРОФИЛЮ «ИНЖЕНЕРНОЕ ДЕЛО»

38557

регистрационный номер

Информационная безопасность и цифровая криминалистика
название секции

Разработка программы для обнаружения и предотвращения угроз и утечек конфиденциальной информации
название работы

Автор: ***Гапуленко Дмитрий Андреевич***
фамилия, имя, отчество
ГБОУ Школа №1155, 10
наименование учебного заведения, класс

Научный руководитель: ***Юсупова Кристина Олеговна***
фамилия, имя, отчество
ГБОУ Школа №1155
место работы
учитель информатики
звание, должность

_____
подпись научного руководителя

Москва – 2024

Разработка программы для обнаружения и предотвращения угроз и утечек конфиденциальной информации

Аннотация

Целью работы является разработка и реализация программы на языке Python, интегрирующей методы машинного обучения и социальной инженерии для обеспечения эффективного обнаружения и предотвращения внутренних угроз и утечек конфиденциальных данных.

Созданная программа способна выявлять аномальное поведение, анализируя большие объемы данных с помощью алгоритмов машинного обучения. Программа предназначена для обнаружения фишинговых угроз в несколько этапов: по тексту сообщения в письме и по сканированию ссылок и файлов, содержащихся внутри письма. Поскольку в реальных условиях, с учетом многообразия и сложности киберугроз, требуется создание комплексных решений, способных адаптироваться и эффективно сопротивляться атакам.

Работоспособность алгоритма обеспечена применением методов социальной инженерии для распознавания попыток манипуляций и социальных атак. При обнаружении потенциальной угрозы, или утечки данных, программа активирует механизмы предупреждения и предотвращения, обеспечивая защиту информационной безопасности.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
ОСНОВНАЯ ЧАСТЬ 5
1 Анализ текста сообщения 5
2 Сканирование файлов и ссылок из письма на наличие вирусов 12
ЗАКЛЮЧЕНИЕ 17
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18
Приложение А. Отзыв 19

ВВЕДЕНИЕ

В современном цифровом мире пользователи сталкиваются с разнообразными киберугрозами, такими как фишинговые письма и вредоносное программное обеспечение. Для эффективной борьбы с этими угрозами было разработано приложение, сочетающее машинное обучение, социальную инженерию и удобный интерфейс. Это позволяет пользователям без глубоких знаний в области информационной безопасности легко защищать себя от атак, автоматически анализируя электронные письма на предмет мошеннических ссылок и вредоносных вложений. Приложение повышает уровень безопасности, делая продвинутые методы защиты доступными каждому, что способствует минимизации рисков утечек данных и инфицирования систем.

В работе была поставлена цель разработать и реализовать программу на языке Python, интегрирующий методы машинного обучения и социальной инженерии для обеспечения эффективного обнаружения и предотвращения внутренних угроз и утечек конфиденциальных данных.

Для достижения поставленной цели были выполнены следующие задачи:

1. изучить современные методы и алгоритмы машинного обучения, применимые для анализа больших объемов данных и выявления аномальных паттернов поведения;
2. разработать алгоритмы обнаружения угроз, основанные на методах машинного обучения, адаптированные для реальных сценариев угроз;
3. интегрировать методы социальной инженерии для распознавания и нейтрализации попыток манипуляций и атак;
4. реализовать программный продукт на языке Python, обеспечивающий автоматизацию процесса обнаружения и предотвращения угроз;
5. провести комплексное тестирование программы в реальных условиях, оценить эффективность и точность обнаружения и предотвращения угроз.

ОСНОВНАЯ ЧАСТЬ

1 Анализ текста сообщения

Для анализа текста сообщений была создана модель, схема которой представлена на рисунке 1.

Рисунок 1 — Схема модели

Как видно из схемы, модель состоит из отдельных частей, имеющих определенный функционал. Импорт необходимых библиотек: в программе используются библиотеки NumPy, scikit-learn для обработки данных, векторизации текста, обучения модели и оценки её эффективности. Функция read_data читает исходные данные (электронные письма) из файла, разделяет их на содержание и метки (например, спам или не спам). Функция read_new_emails читает новые электронные письма из файла, предназначенные для классификации моделью. Функция classify_new_emails принимает обученную модель и векторизатор классифицирует новые электронные письма. На этапе предварительной обработки данных используется TfidfVectorizer для преобразования текста писем в числовые векторы, которые могут быть использованы для обучения модели. В результате обучения модели данные

разделяются на обучающую и тестовую выборки. Используется наивный байесовский классификатор (MultinomialNB) для обучения на обучающей выборке. Далее происходит тестирование модели: модель тестируется на тестовой выборке, и выводится отчет о классификации, показывающий эффективность модели. После успешного тестирования программой выполняется классификация новых писем, считанных из файла. Результаты классификации выводятся, показывая, как каждое письмо было классифицировано моделью.

На рисунке 2 представлен код программы «main.py», реализующий весь перечисленный выше функционал: обучение и тестирование модели, а также определение фишинговых писем среди нового набора данных.

```python
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# Функция для чтения исходных данных
def read_data(filename):
    emails = []
    labels = []
    with open(filename, 'r', encoding='utf-8') as file:
        for line in file:
            if ' 1' in line or ' 0' in line or ' 2' in line:
                content, label = line.rsplit(' ', 1)
                emails.append(content.strip())
                labels.append(int(label.strip()))
    return emails, labels

# Функция для чтения новых писем из файла
def read_new_emails(filename):
    new_emails = []
    with open(filename, 'r', encoding='utf-8') as file:
        new_emails = [line.strip() for line in file]
    return new_emails

# Функция для классификации новых писем
def classify_new_emails(model, vectorizer, new_emails):
    new_emails_vectorized = vectorizer.transform(new_emails)
    predictions = model.predict(new_emails_vectorized)
    return predictions
```

```python
# Чтение и предварительная обработка данных
emails, labels = read_data('emails.txt')
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(emails)

# Разделение и обучение модели
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.3,
stratify=labels, random_state=42)
model = MultinomialNB()
model.fit(X_train, y_train)

# Тестирование модели
predictions = model.predict(X_test)
print(classification_report(y_test, predictions, zero_division=1))

# Чтение и классификация новых писем из файла
new_emails_file = 'new_emails.txt'
new_emails = read_new_emails(new_emails_file)
new_predictions = classify_new_emails(model, vectorizer, new_emails)

# вывод результатов
for email, prediction in zip(new_emails, new_predictions):
    classification = 'Phishing' if prediction == 1 else 'Normal' if prediction == 0 #else
'Unnecessary'
    print(f"Email: {email}\nClassified as: {classification}\n")
```

Рисунок 2 — Код программы

Опишем этап обучения модели. Машина обучалась на входных данных в виде текстовых писем и классифицировалась числами 1 и 0, которые

обозначают: 0 – нормальное сообщение, которое не несет угрозы; 1 – фишинговое сообщение, которое несет угрозу для пользователя.

Для обучения машины было интегрировано чтение текста из отдельного файла (рис. 3), в который заносились почтовые письма в виде:

«(Номер сообщения). (Текст сообщения). (Классификация 1 или 0)».

Примеры:

фишинговое – «31. Подтвердите свою учетную запись, перейдя по следующей ссылке. 1»;

нормальное – «32. Добрый день! Спасибо за ваш выбор нашей компании. 0».

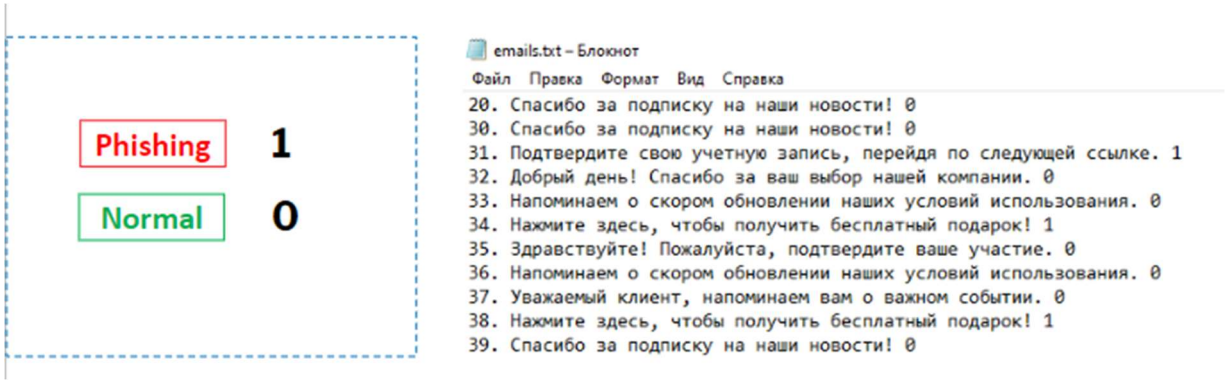

Рисунок 3 — Обучающая выборка

Далее после завершения обучения программа выводит итоговый результат.

	precision	recall	f1-score	support
0	0.93	1.00	0.96	1267
1	1.00	0.93	0.96	1267
accuracy			0.96	2534
macro avg	0.97	0.96	0.96	2534
weighted avg	0.97	0.96	0.96	2534

Рисунок 4 — Вывод программы результатов обучения модели

На рисунке 4 представлены расчетные характеристики, по которым можно сделать вывод о качестве обучения. Разберем эти показатели.

- Precision (точность). Доля правильно идентифицированных положительных результатов из всех результатов, которые модель идентифицировала как положительные. Например, точность 1.00 для класса 1 означает, что, когда модель предсказывает класс 1, она правильна в 100% случаев.
- Recall (полнота). Доля правильно идентифицированных положительных результатов из всех реальных положительных результатов. Например, полнота 0.93 для класса 1 означает, что модель правильно идентифицировала 93% всех реальных случаев класса 1.
- F1-score. Среднее гармоническое точности и полноты. Это мера, которая учитывает и точность, и полноту, чтобы дать общую оценку качества модели. F1-score равный 0.96 для класса 1 указывает на высокую общую эффективность модели в классификации этого класса.
- Support. Количество реальных случаев в каждом классе. Например, в классе 0 и классе 1 по 27 случаев.
- Accuracy (точность). Общая доля правильных предсказаний из всех предсказаний. Значение 0.96 означает, что 96% предсказаний модели были верными.
- Macro avg. Среднее значение точности, полноты и F1-score, взятое по всем классам, без учета дисбаланса классов.
- Weighted avg. Средневзвешенное значение точности, полноты и F1-score, где веса соответствуют количеству случаев в каждом классе.

После завершения этапа обучения, когда по показателям мы убедились, что обучение прошло успешно, далее следует тестирование модели. Для этого машине подается набор новых писем, записанных в текстовом файле «new_emails.txt», письма при этом не классифицированы, так как машина должна сама определить, где фишинг, а где нормальное сообщение. В программе реализовано чтение новых писем из файла и определение, к какому классу они относятся.

Например, подадим машине десять новых писем следующего содержания.

- «Поздравляем! Ваша учетная запись была выбрана для участия в эксклюзивной программе лояльности. Чтобы активировать свои привилегии, перейдите по ссылке и введите личные данные. Остерегайтесь подделок - используйте только компьютер.»
- «Уважаемый клиент, мы обнаружили подозрительные попытки входа в ваш аккаунт. Для вашей безопасности, перейдите по прикрепленной ссылке и введите свои данные для подтверждения. Не откладывайте, ваша безопасность на первом месте.»
- «Ваш банковский аккаунт требует срочного обновления. Для предотвращения блокировки, перейдите по ссылке и обновите свои данные. Внимание: делайте это только с компьютера для обеспечения безопасности.»
- «Здравствуйте! Вы были выбраны для участия в эксклюзивном опросе с шансом выиграть ценные призы. Для участия перейдите по ссылке и заполните форму. Рекомендуется использовать компьютер для оптимального опыта.»
- «Оповещение: Ваша учетная запись электронной почты скоро будет деактивирована. Чтобы предотвратить это, незамедлительно подтвердите свои данные, перейдя по ссылке. Используйте компьютер для доступа к форме.
- «Добрый день! Прилагаю к этому письму документы к нашему следующему проекту. Пожалуйста, ознакомьтесь и дайте свои замечания до понедельника. Спасибо.»
- «Уважаемые коллеги, напоминаю о совещании завтра в 10 утра. Мы будем обсуждать график и бюджет предстоящих мероприятий. С нетерпением жду ваших идей и предложений.»
- «Привет! Как твои дела? Хочу пригласить тебя на кофе в это воскресенье. Будет приятно поболтать и обсудить наши последние новости. Дай знать, удобно ли тебе.»

- «Здравствуйте, это напоминание о вашем предстоящем визите к врачу в следующий четверг в 15:00. Пожалуйста, подтвердите ваше присутствие. С уважением, клиника «Здоровье'».»
- «Дорогой друг, благодарю за прекрасный вечер вчера! Фотографии получились замечательные. Я отправлю их тебе в следующем письме. До скорого!»

На рисунке 5 показан вывод программой тестовой выборки писем с присвоенной им меткой: «Normal» – нормальное сообщение, то есть сообщение под классом 0, «Phishing» – фишинговое сообщение, то есть сообщение под классом 1.

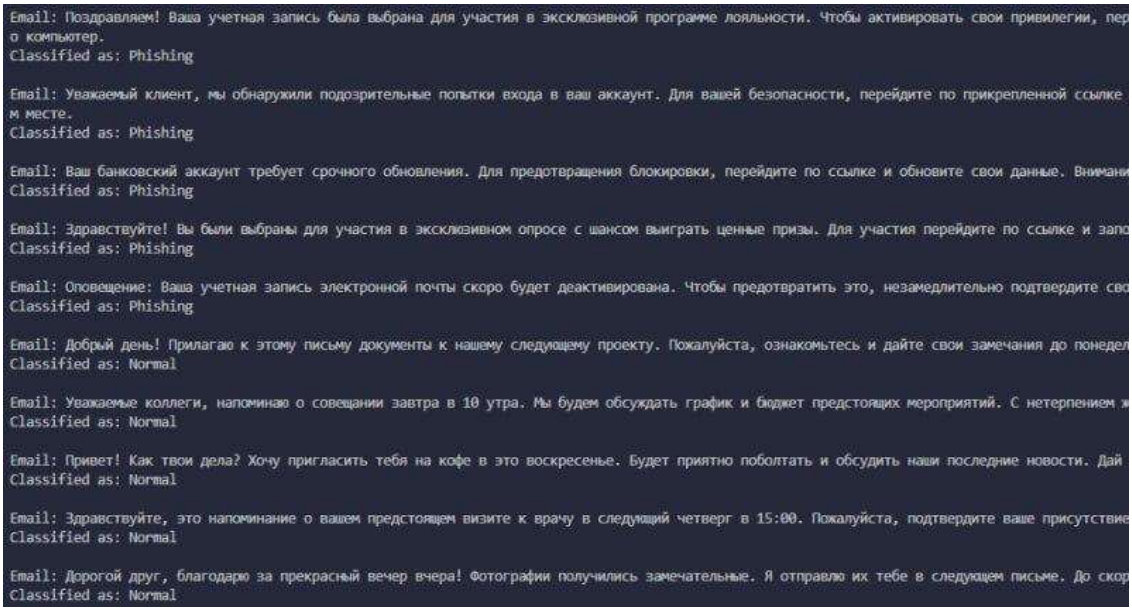

Рисунок 5 — Результат классификации писем на этапе тестирования

Проведя анализ результатов, определено, что модель все тестовые письма отнесла к верному классу: верно выявила фишинговые тексты сообщений. Данный факт был подтвержден и на тестовых данных большего размера. Это позволяет сделать вывод о работоспособности модели: программа с высокой точностью верно классифицирует письма на фишинговые и нормальные, что позволяет доверять результатам и применять данный алгоритм на практике для анализа текстов входящих сообщений.

2 Сканирование файлов и ссылок из письма на наличие вирусов

Следующее расширение реализуемой модели – это проверка письма на наличие в нем ссылок и вложенных файлов и их дальнейшее сканирование на вирусы и попытку кражи данных. В ситуации, когда анализ текста сообщения не выявил подозрительное содержание и не определил письмо как фишинговое, необходима вторичная обработка сообщения на скрытые угрозы, с которыми пользователь может столкнуться, пройдя по ссылке в письме или открыв вложенный файл. Такая вторичная проверка письма была также реализована в работе в программе «pyparser.py», структуру которой опишем подробно.

```
import imaplib
import email
import os
import time
import re
import requests
import logging
from email.header import decode_header
from bs4 import BeautifulSoup

# Импорт функций из main.py
from main import read_new_emails, classify_new_emails, vectorizer, mode
```

Рисунок 6 — Подключение библиотек

В начале программного файла подключаются необходимые библиотеки для работы с почтовыми серверами анализа и обработки электронных писем, работы с HTML-контентом, ведения логов и взаимодействия с внешними сервисами (рис. 6). Также импортируются функции из другого файла «main.py», предназначенные для чтения, классификации писем и работы с моделью машинного обучения.

```
# Конфигурация
first_run = True
email_address = "" #почту
password = "" #пароль приложения
api_key = "" #апи вирустотал
output_file_path = "" #файл куда сохранять письма
```

Рисунок 7 – Задание конфигурационных параметров

На рисунке 7 показано, как задаются основные конфигурационные параметры, включая учетные данные почтового ящика и API, путь к файлу для сохранения обработанных данных. Переменная first_run используется для контроля поведения скрипта при первом запуске.

```
def clean_filename(filename):
    # Декодирование, если имя файла закодировано
    decoded_string, encoding = decode_header(filename)[0]
    if encoding:
        filename = decoded_string.decode(encoding)

    # Удаление или замена недопустимых символов
    filename = re.sub(r'[\\/*?:"<>|]', '_', filename)
    return filename
```

Рисунок 8 – Функция очистки и декодирования имен файлов вложений

Функция clean_filename предназначена для очистки и декодирования имен файлов вложений, удаления или замены недопустимых символов, чтобы предотвратить возможные ошибки при сохранении файлов на диск (рис. 8).

```
def extract_urls(email_body):
    link_pattern = re.compile(r'http[s]?://\S+')
    return link_pattern.findall(email_body)
```

Рисунок 9 – Функция извлечения всех URL из текста письма

Функция extract_urls предназначена для извлечения всех URL из текста электронного письма с использованием регулярных выражений (рис. 9). Это позволяет обнаружить и анализировать ссылки в письмах, что может быть полезно для выявления фишинговых атак.

```
def extract_text_from_html(html_content):
    soup = BeautifulSoup(html_content, 'html.parser')
    text = soup.get_text(separator=' ', strip=True)
    return text
```

Рисунок 10 – Функция преобразования HTML-контента в чистый текст

Функция extract_text_from_html использует библиотеку BeautifulSoup для преобразования HTML-контента письма в чистый текст, удаляя все HTML-теги

и оставляя только текстовое содержимое. Это упрощает анализ содержания писем.

```
def process_email(email_id, mail, output_file_path, api_key,
email_address, password):
    …
```

Рисунок 11 – Функция извлечения и обработки содержимого письма

Основная функция process_email отвечает за обработку каждого письма. Она извлекает текст письма, обрабатывает его содержимое, классифицирует, сканирует URL и вложения с помощью VirusTotal, и сохраняет результаты. Это ключевая функция всей программы, отвечающая за основную логику обработки.

```
def scan_url_with_virustotal(api_key, url):
…
def scan_file_with_virustotal(api_key, file_path):
    …
```

Рисунок 12 – Функция запуска постоянной проверки почтового ящика

В конце файла находится точка входа в программу, где вызывается функция main_loop. Эта функция запускает бесконечный цикл обработки новых писем, обеспечивая постоянную проверку почтового ящика на наличие новых сообщений.

Для тестирования работоспособности данного функционала и демонстрации результатов на почту было отправлено шесть новых писем, которые прошли обработку созданным алгоритмом. Первым шагом был проверен текст сообщения. По результатам проверки содержащегося в нем текста сообщение было классифицировано: фишинг или нет. Далее отдельно была произведена проверка вложенных файлов на наличие угроз и просканирован адрес указанного в письме ресурса (при наличии). На рисунке представлен отчет, который выводит программа после обработки каждого письма.

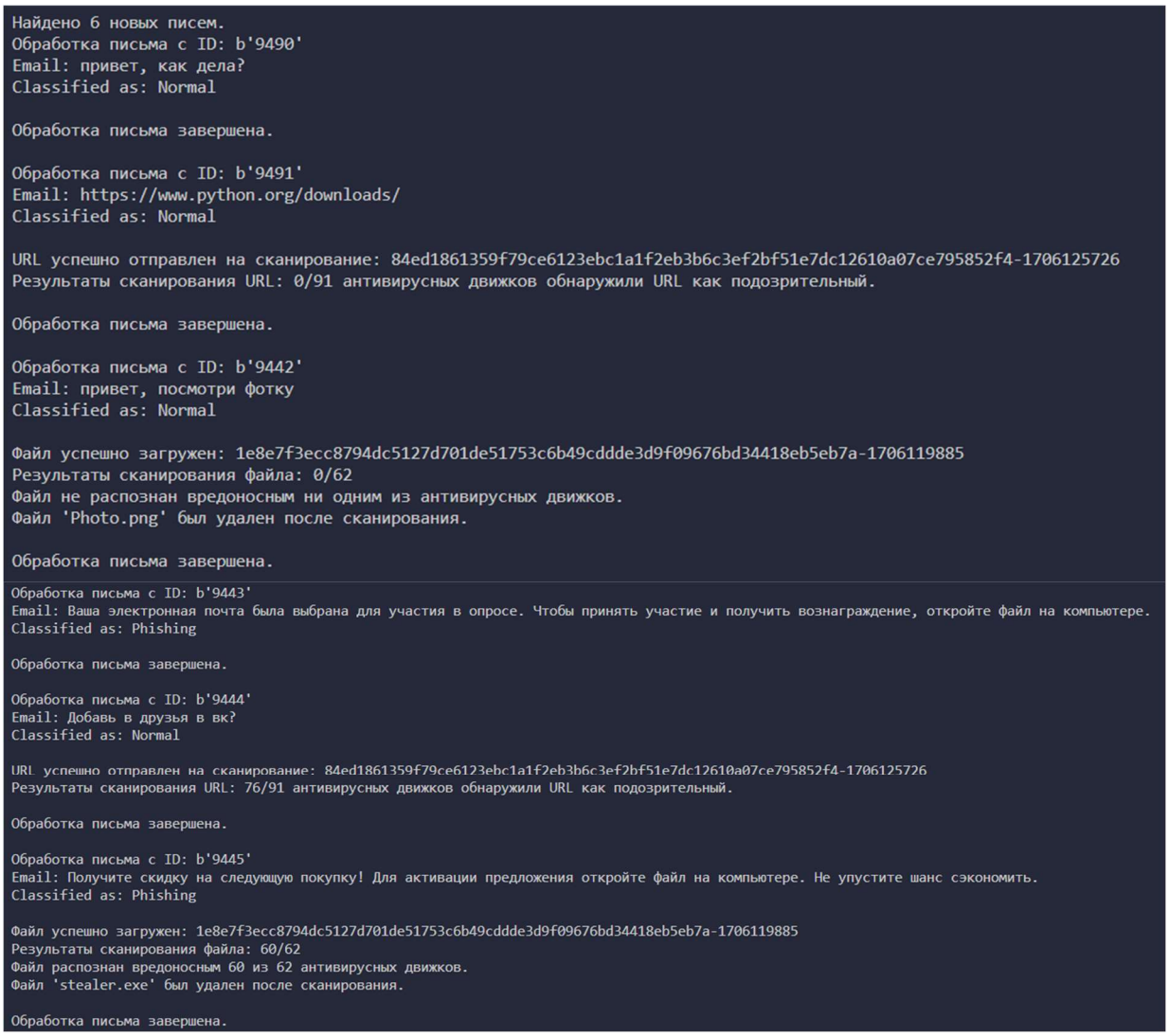

Рисунок 13 – Результат полной обработки новых писем на наличие угроз

Данные алгоритмы проверки были собраны в приложение, которое также написано на языке Python с помощью библиотеки PyQt 5 версии (рис. 14). Приложение включает в себя механизм самообучения, который способствует постоянному улучшению качества обнаружения фишинга и вредоносных атак. После анализа каждого сообщения программа обращается к пользователю с запросом оценить правильность классификации письма – фишинговое или нормальное. Ответ пользователя используется для дополнения обучающего набора данных: письмо с соответствующей меткой класса заносится в текстовый документ, который впоследствии применяется для дополнительной тренировки модели машинного обучения. Этот процесс не только повышает точность распознавания угроз, но и адаптирует систему под специфику общения

конкретного пользователя, обеспечивая более персонализированную и эффективную защиту.

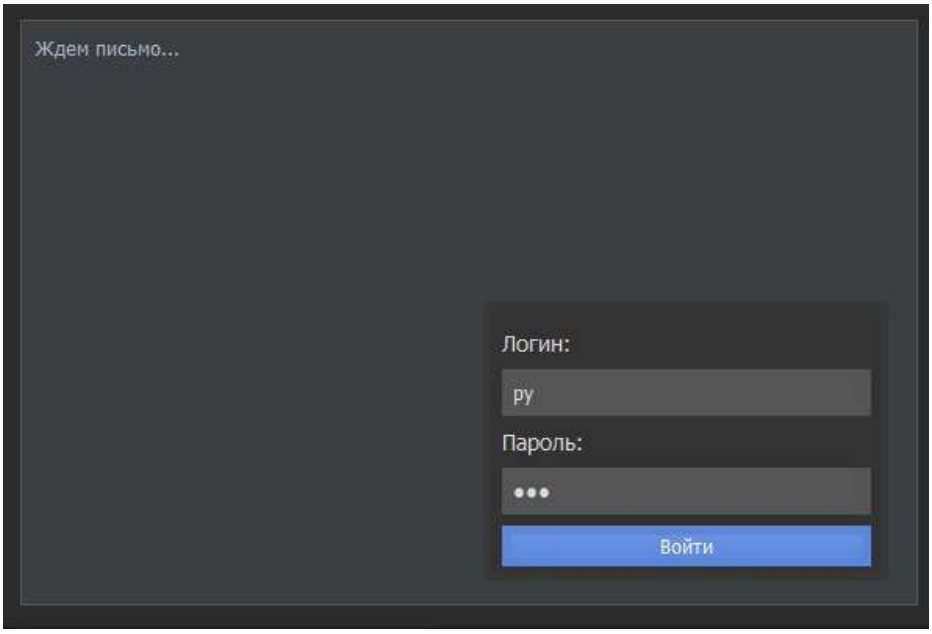

Рисунок 14 – Фрагмент интерфейса приложения

На рисунке 14 представлено окно входа в программу, для которого используется логин и пароль от почты. После авторизации запускаются алгоритмы сканирования входящих писем, после чего выводятся результаты по оценке письма на фишинг и проверке ссылок и файлов на вирусы и угрозы.

ЗАКЛЮЧЕНИЕ

В рамках данного проекта реализован очень актуальный в настоящее время инструмент, способный распознать фишинговые письма. Действительно, большинство фишинговых атак осуществляется именно через электронную почту. Результативность разработанных алгоритмов обусловлена способностью распознать даже самые ложные многоуровневые фишинговые атаки.

Созданная программа в активном состоянии способна непрерывно анализировать почтовый ящик на наличие новых писем. Поступившие письма далее обрабатываются на предмет содержания подозрительного текста сообщения. Если же текст проверку прошел не успешно, то письмо алгоритмом классифицируется как фишинговое. Кроме этого, сообщение проверяется на содержание ссылок и прикрепленных файлов. При их присутствии машина производит и их сканирование на наличие угроз.

Такой подход в поэтапной проверке вновь поступившей почты защитит пользователя от утечки данных и заражения вирусами. Процесс обработки писем программой производится достаточно быстро, поэтому созданные алгоритмы сканирования сообщений были объединены в удобное для пользователя приложение, обеспечивающие ему безопасность.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. П.А. Карасев. Информационная безопасность в корпоративных сетях [Электронный ресурс] URL: https://cyberleninka.ru/article/n/informatsionnaya-bezopasnost-v-korporativnyh-setyah?ysclid=lsm79qnhhx498076160 (дата обращения: 3.12.2023)
2. Д.И. Пирштук. Машинное обучение и анализ данных [Электронный ресурс] URL: https://elib.bsu.by/handle/123456789/270017?mode=full&ysclid=lsm76fj3lz253183771 (дата обращения: 14.11.2023)
3. Ю.И. Химонин. Сбор и анализ требований к программному продукту [Электронный ресурс] URL: https://studfile.net/preview/3066071/ (дата обращения: 12.01.2024)

Приложение А. Отзыв

Федеральное государственное
бюджетное образовательное учреждение
высшего образования
«Национальный исследовательский
университет «МЭИ» (ФГБОУ «МЭИ»)
111250, г. Москва,
вн.тер.г. муниципальный округ Лефортово,
ул. Красноказарменная, д. 14, стр. 1
Тел.: (495) 362-75-60, факс: (495) 362-89-38
E-mail: universe@mpei.ac.ru
http://www.mpei.ac.ru

№ _____

«___» _____ 20 __ г.

ОТЗЫВ

на **проектную** работу
(проектную / исследовательскую)

по теме **«Разработка программы для обнаружения и предотвращения угроз и утечек конфиденциальной информации»**
(наименование работы)

учащегося **___10_____** класса _____ **ГБОУ Школа №1155** _____
(класс) *(наименование образовательной организации)*

_____ **Гапуленко Дмитрий Андреевич** _____.
(фамилия, имя, отчество (при наличии), заполняется на каждого участника)

В проекте «Разработка программы для обнаружения и предотвращения угроз и утечек конфиденциальной информации» автором создан очень полезный и актуальный в настоящее время инструмент, который способен помочь пользователю в обнаружении фишинговых угроз.

В данной работе автор создал программу, интегрирующую методы машинного обучения и социальной инженерии для обеспечения эффективного обнаружения и предотвращения внутренних угроз и утечек конфиденциальных данных. В проекте наглядно продемонстрированы этапы обучения и тестирования модели, представлены результаты определения фишинговых писем с высокой точностью, что подтверждает эффективность реализованного подхода. Кроме того, автором создан алгоритм, который сканирует ссылки и файлы внутри письма на наличие угроз. В проекте приводится подробная качественная схема структуры программы, а также полное описание ее реализации.

Цели работы автором достигнуты, актуальность представленных в работе достижений и их практическая значимость в области информационной безопасности автором подтверждены. Разработанный в рамках проекта продукт способен предупредить и современные фишинговые угрозы с различными сложными тактиками, для этого автором дополнительно настроено самообучение модели. Стоит также отметить, что весь изложенный в работе функционал в конечном виде представляет собой приложение, работоспособность которого также наглядно продемонстрирована.

доцент — *(должность)*

(подпись)

Пямасов А.К. — *(И. О. Фамилия)*

М. П.