Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Симонова Олена Михайлівна,
група РЗ-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Захист інформаційно-телекомунікаційних мереж від шкідливого програмного забезпечення в контексті сучасних гібридних війн

Спеціальність:
125 Кібербезпека
Спеціалізація, освітня програма Кібербезпека

Керівник:
Бобок Іван Ігорович,
д.т.н., доцент

Одеса – 2022

Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність 125 Кібербезпека
Спеціалізація, освітня програма Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Симоновій Олені Михайлівні

1.Тема роботи: *Захист інформаційно-телекомунікаційних мереж від шкідливого програмного забезпечення в контексті сучасних гібридних війн*,
керівник роботи *Бобок Іван Ігорович, д.т.н., доцент*
затверджені наказом ректора від „____”_____ 20___ р. №_____ .
2.Зміст роботи*: роль фішингових листів відомих кібератак, опис алгоритма виявлення фішингових електронних листів, програмна реалізація алгоритма виявлення фішингових електронних листів*
3. Перелік ілюстративного матеріалу: *Отриманий лист під час атаки Petya, Статистика заражених організацій по країнам, Розміщений Службою спеціального зв'язку та захисту інформації України приклад листа, Модель URLNet, Блок-схема перевірки Return-Path, Блок-схема перевірки Reply-To, Блок-схема перевірки Received-SPF, Блок-схема перевірки DKIM, Блок-схема перевірки листа за словником спаму, Блок-схема перевірки листа за словником тональності, Блок-схема перевірки URL-посилань на кількість символів «/», Блок-*

схема перевірки URL-посилань на загальну кількість символів, Блок-схема перевірки URL-посилань на наявність символу равлика, , Статистика популярності браузерів у світі, Статистика популярності браузерів на ПК в Україні, Демонстрація інтерфейсу програми в різних варіаціях, Архітектура розширення Chrome, Архітектура розширення Chrome – Athena logic, Фрагме сторінки інструкції, , Приклад листа, використаного під час дослідження

4. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури за темою випускної кваліфікаційної роботи*	*02.09.2022*	*виконано*
2	*Розробка алгоритму захисту електронної пошти, використовуючи гібридний метод аналізу*	*19.09.2022*	*виконано*
3	*Розробка програмної реалізації запропонованого алгоритму*	*20.10.2022*	*виконано*
4	*Підготовка тексту роботи*	*14.11.2022*	*виконано*
5	*Підготовка презентації та доповіді*	*28.11.2022*	*виконано*
7	*Попередній захист*	*02.12.2022*	*виконано*
8	*Нормоконтроль, рецензування*	*18.12.2022*	*виконано*
9	*Занесення роботи в електронний архів*	*18.12.2022*	*виконано*
10	*Допуск до захисту*	*18.12.2022*	*виконано*

Здобувач вищої освіти _____ *Симонова О.М.*

Керівник роботи _____ *Бобок І.І.*

АНОТАЦІЯ

Кваліфікаційна робота на тему «Захист інформаційно-телекомунікаційних мереж від шкідливого програмного забезпечення в контексті сучасних гібридних війн» на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 20 рисунків, 5 таблиць, 2 додатки, 48 літературних джерел за переліком посилань. Робота виконана на 73 сторінках загального тексту і 49 сторінках основного тексту.

Метою роботи є розробка алгоритму та програмної реалізації захисту пошти від фішингових атак, яка може бути імплантована у сучасний браузер.

Об’єктом дослідження роботи є процес забезпечення безпеки інформаційно-телекомунікаційних мереж від фішінгових атак. Предметом – методи ідентифікації та попередження фішингових атак.

Наукова новизна отриманих результатів полягає в наступному:

Вперше розроблено алгоритм виявлення небезпечних електронних листів, який відрізняється від існуючих комплексним підходом до аналізу і використанням раніше не застосованих у цьому напрямку методів, що дозволило значно підвищити ефективність аналізу.

Практичне значення отриманих результатів полягає в розробці алгоритму захисту пошти від фішингових атак, яка може бути імплантована у сучасний веб-браузер. Результати роботи можуть бути використані для захисту інформаційно-телекомунікаційних мереж від розповсюдження шкідливого програмного забезпечення та/або витоку даних.

ІНФОРМАЦІЙНА БЕЗПЕКА, КІБЕРБЕЗПЕКА, ФІШИНГ, ЕЛЕКТРОННА ПОШТА

ANNOTATION

Qualification work on the topic "Protection of information and telecommunications networks from malicious software in the context of modern hybrid wars" for obtaining the second (master's) level of higher education in the specialty 125 Cybersecurity, specialization, educational program: Cybersecurity, contains 20 figures, 5 tables, 2 appendices , 48 literary sources according to the list of references. The work was completed on 73 pages of the general text and 49 pages of the main text.

The purpose of the work is to develop an algorithm and software implementation of mail protection against phishing attacks, which can be implanted in a modern browser.

The object of research is the process of ensuring the security of information and telecommunication networks against phishing attacks. The subject is methods of identification and prevention of phishing attacks.

The scientific novelty of the obtained results is as follows:

For the first time, an algorithm for detecting dangerous e-mails was developed, which differs from the existing ones by a comprehensive approach to analysis and the use of methods not previously applied in this direction, which allowed to significantly increase the efficiency of the analysis.

The practical significance of the obtained results lies in the development of an algorithm to protect mail from phishing attacks, which can be implanted in a modern web browser. The results of the work can be used to protect information and telecommunication networks from the spread of malicious software and/or data leakage.

INFORMATION SECURITY, CYBER SECURITY, PHISHING, EMAIL

ЗМІСТ

ВСТУП

Зі значним зростанням використання Інтернету люди все частіше діляться своїми персональними даними в мережі. Як наслідок, величезна кількість персональних даних та фінансових операцій стають вразливими для кіберзлочинців.

Актуальність теми, насамперед, полягає у тому, що, незважаючи на існування багатьох методів та підходів для захисту користувачів від фішинг-атак [1-4], питання захисту у цьому напрямку залишається доволі гострим.

За даними Національного координаційного центру кібербезпеки, у 2021 році в Україні зафіксовано понад 400 000 випадків фішингових атак [5]

Для боротьби з фішингом поштові клієнти використовують спам-фільтри, які відправляють підозрілі листи в карантин (папки спаму), а не в основну поштову скриньку користувача. Однак ці фільтри не завжди ефективні: з понад 555 000 фішингових листів, проаналізованих компанією хмарної безпеки Avanan в рамках свого Global Phish Report 2019 [6], 25% оминули заходи безпеки Office 365, в результаті чого потрапили до поштових скриньок потенційних жертв.

Фішинг використовується для розповсюдження шкідливого програмного забезпечення і зараження мереж критичної інфраструктури. Прикладами використання електронної пошти для фішингової атаки є розповсюдження вірусу Petya та NotPetya у 2017 році, який брав свій початок з української державної енергокомпанії.

Метою роботи є розробка алгоритму та програмної реалізації захисту пошти від фішингових атак, яка може бути імплантована у сучасний браузер.

Для досягнення визначеної мети в магістерській атестаційній роботі були сформовані для вирішення наступні задачі:

– аналіз існуючих методів з забезпечення безпеки інформаційно-телекомунікаційних мереж від фішингових атак;

– розробка алгоритму з використанням комплексного методу з забезпечення безпеки інформаційно-телекомунікаційних мереж;

– розробка програмної реалізації запропонованого алгоритму;

– дослідження ефективності запропонованого алгоритму шляхом порівняння з іншими.

Об’єктом дослідження роботи є процес забезпечення безпеки інформаційно-телекомунікаційних мереж від фішінгових атак. Предметом – методи ідентифікації та попередження фішингових атак.

1. РОЛЬ ФІШИНГОВИХ ЛИСТІВ ВІДОМИХ КІБЕРАТАК

1.1 Доведення актуальності теми на відомих прикладах фішингових атак

Фішинг – це різновид соціальної інженерії, коли зловмисник надсилає шахрайське (наприклад, підроблене чи іншим чином оманливе) повідомлення, призначене для того, щоб оманою змусити людину розкрити конфіденційну інформацію зловмисникові [7] або розгорнути шкідливе програмне забезпечення в інфраструктурі жертви, як-от програми-вимагачі. Фішинг є прикладом високоефективної форми кіберзлочинності [8].

Електронна пошта стала одним із надійних засобів зв’язку в реальному часі, за допомогою якого величезна кількість людей і організацій обмінюються своїми повідомленнями та даними. Відповідно з Radicati Group [9], загальна кількість користувачів електронної пошти становила приблизно 4,1 мільярда на початку 2021 року, і, за оцінками, зросте до 4,5 мільярдів до кінця 2025 року.

Зі значним збільшенням кількості користувачів електронної пошти зловмисники використовують електронну пошту різними способами, щоб спонукати користувачів розкривати свої облікові дані. Наприклад, використовуючи повідомлення електронної пошти з переконливим вмістом як приманку для викрадення особистої інформації користувачів, електронний лист направляє користувача через гіперпосилання на веб-сайт, що належить злочинцям, який дуже схожий на оригінальний веб-сайт. Потім користувача попросять ввести особисту та фінансову інформацію. Насправді це дозволяє злочинцям отримати доступ до цієї цінної інформації, яку вони потім використовують для вчинення шахрайства або продажу. Кіберзлочинці також можуть обманом змусити користувачів завантажити шкідливі коди або зловмисне програмне забезпечення після того, як вони натиснуть на посилання, вбудоване в електронний лист.

Напади можуть мати руйнівні наслідки. Для фізичних осіб це включає несанкціоновані покупки, крадіжки коштів або крадіжки особистих даних. Крім того, фішинг часто використовується для проникнення в корпоративні чи урядові мережі в рамках більшої атаки, наприклад ескалації постійної загрози. В останньому випадку співробітники проникають в обхід периметрів безпеки, розповсюджують зловмисне програмне забезпечення в закритих середовищах або отримують привілейований доступ до захищених даних.

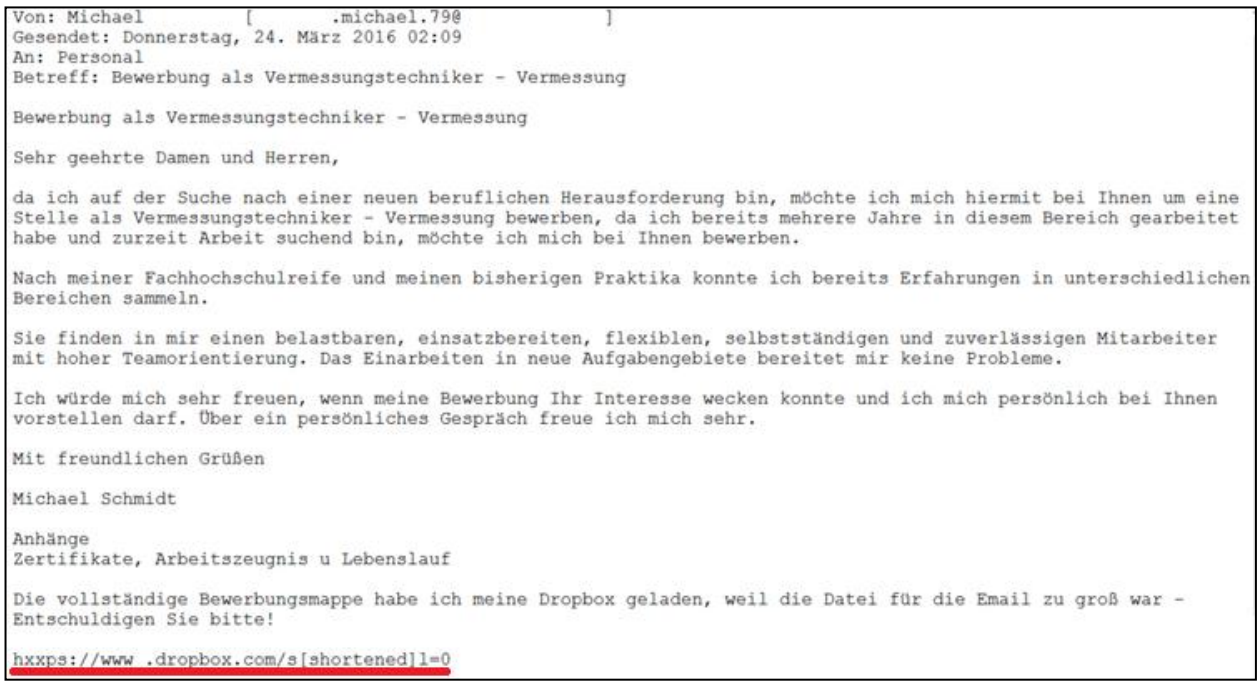
Організації, які потерпають від таких атак, часто зазнають серйозних фінансових втрат на додаток до втрати частки ринку, репутації та довіри споживачів. Залежно від масштабу фішинг може перетворитися на інцидент із безпекою, та компанії може бути важко уникнути наслідків.

Починаючи з 2000 років фішинг набуває популярності у кіберзлочинців. Одну з перших великих, хоча й невдалих, спроб було зроблено 2001 року, коли в хаосі після терактів 11 вересня зловмисники відправили своїм жертвам електронні листи нібито для перевірки їхньої особистості. Отримані дані було використано для крадіжки банківських даних.

З того часу кількість фішингових атак лише збільшується. За даними звіту APWG у другому кварталі 2022 року зафіксовано 1 097 811 фішингових атак, що є новим рекордом і найгіршим кварталом щодо фішингу за всю історію APWG [10].

Одним з прикладів використання електронної пошти для фішингової атаки є розповсюдження вірусу Petya у 2016 році. Це шкідливе програмне забезпечення поширювалось через електроні листи (Рисунок 1.1), та, ймовірно, було розроблено спеціально для атак на організації та підприємства, бо адресувалося відділам, фахівцям кадрів та тим, хто займався переглядом резюме.

Листи приходили нібито від претендентів на вакансію та містили посилання на Dropbox, де можна було завантажити "резюме". Але замість резюме за посиланням завантажувався шкідливий EXE-файл, при запуску якого система аварійно завершувала роботу, появлявся синій екран, перезавантажувався комп’ютер, та всі дані на жорстких дисках ставали зашифрованим.



Von: Michael [.michael.79@]
Gesendet: Donnerstag, 24. März 2016 02:09
An: Personal
Betreff: Bewerbung als Vermessungstechniker - Vermessung

Bewerbung als Vermessungstechniker - Vermessung

Sehr geehrte Damen und Herren,

da ich auf der Suche nach einer neuen beruflichen Herausforderung bin, möchte ich mich hiermit bei Ihnen um eine Stelle als Vermessungstechniker - Vermessung bewerben, da ich bereits mehrere Jahre in diesem Bereich gearbeitet habe und zurzeit Arbeit suchend bin, möchte ich mich bei Ihnen bewerben.

Nach meiner Fachhochschulreife und meinen bisherigen Praktika konnte ich bereits Erfahrungen in unterschiedlichen Bereichen sammeln.

Sie finden in mir einen belastbaren, einsatzbereiten, flexiblen, selbstständigen und zuverlässigen Mitarbeiter mit hoher Teamorientierung. Das Einarbeiten in neue Aufgabengebiete bereitet mir keine Probleme.

Ich würde mich sehr freuen, wenn meine Bewerbung Ihr Interesse wecken konnte und ich mich persönlich bei Ihnen vorstellen darf. Über ein persönliches Gespräch freue ich mich sehr.

Mit freundlichen Grüßen

Michael Schmidt

Anhänge
Zertifikate, Arbeitszeugnis u Lebenslauf

Die vollständige Bewerbungsmappe habe ich meine Dropbox geladen, weil die Datei für die Email zu groß war - Entschuldigen Sie bitte!

hxxps://www .dropbox.com/s[shortened]l=0

Рисунок 1.1 – Отриманий лист під час атаки Petya

27 червня 2017 року почалася масштабна глобальна кібератака із застосуванням нового варіанту Petya. Ця програма-вимагач є модифікованою версією Petya, яка називається NotPetya, щоб відрізнити цю атаку від старої версії.

NotPetya відрізняється від старих версій високим рівнем шифрування, яке шифрує не тільки файли, але й всю систему. Він шифрує головну файлову таблицю (MFT) після перезавантаження зараженої системи, тому головний завантажувальний запис (MBR) стає непрактичним. У результаті, блокуючи MBR, заражена система зрештою стає марною, тому стає неможливим отримати доступ до файлів або навіть операційної системи на диску, оскільки MBR, який є сектором жорсткого диска, важливий для ідентифікації розташування операційної системи та файлів. NotPetya поширюється, використовуючи переваги EternalBlue– вразливістю в операційній системі Windows. Атака NotPetya може отримувати інформацію про доступ адміністратора на одному невиправленому комп’ютері в мережі та поширюватись на інші комп’ютери в тій самій мережі.

Одними з перших про проблеми повідомили українські компанії, зокрема державна енергетична компанія та головний аеропорт Києва [11]. Один зі

способів розповсюдження атаки – фішингові електронні листи, які містили вкладення зі зловмисним програмним забезпеченням [12].

Ця атака стосувалася не тільки України, вона також зафіксована в інших 64 країнах світу, включаючи Європу та США [13]. Виходячи з рисунка 1.2, США є другою після України країною, ураженою NotPetya.

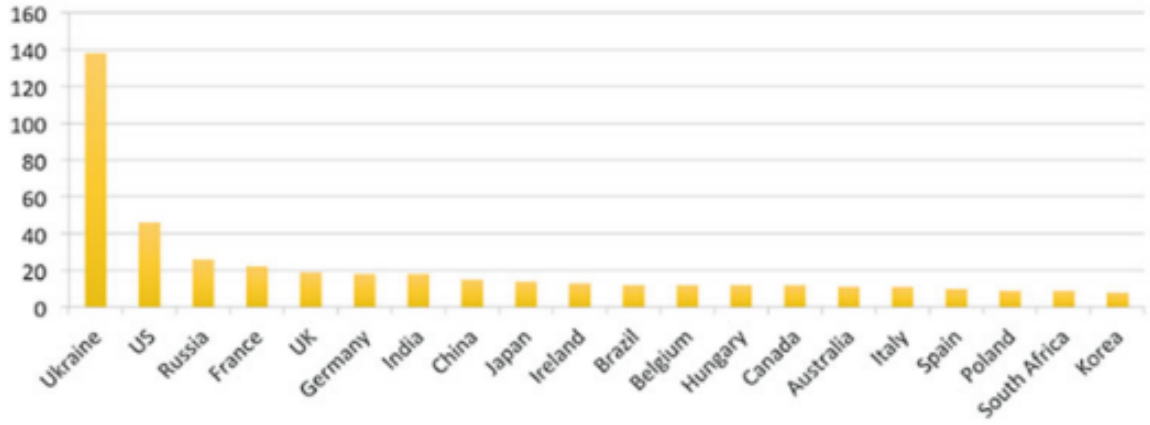


Рисунок 1.2 – Статистика заражених організацій по країнам

У звіті, опублікованому Wired [14], за оцінкою Білого дому загальні збитки, завдані NotPetya, перевищують 10 мільярдів доларів. Це підтвердив колишній радник з внутрішньої безпеки Том Боссерт , який на момент атаки був найвищим посадовцем з кібербезпеки в уряді США. Україна, в свою чергу, втратила понад 10 млрд гривень [15].

Ця кібератака є часиною кібервійни та складовою російсько-української війни. Відповідальність за кібератаку на Росію поклали всі п'ять країн-членів союзу FVEY: Австралія, Велика Британія, Канада, Нова Зеландія, Сполучені Штати, а також уряди Данії та України [16].

Приклад розповсюдження першого варіанту Petya належить до типу масової фішинг-атаки. Характерною ознакою цього типу є «сліпа спроба» поширення вірусу-вимагача: листи приходили на e-mail тим, хто відповідав за пошук кадрів і це було єдиним, що пов’язувало цих людей. Згідно з цим типом атаки,

зловмисник ставить на мету поширити якомога більше фішингових листів, й не концентрується на конкретній жертві.

Другий же розглянутий варіант – NotPetya – належить до цільового фішингу. Цільовий фішинг відрізняється від масового тим, що він спрямований на конкретну мету і є більш небезпечним, оскільки кіберзлочинці спеціально збирають інформацію про жертву, щоб зробити повідомлення більш переконливим. Добре складений цільовий фішинговий лист дуже важко відрізнити від законного, не шкідливого листа. Цільовий фішинг – один із найефективніших методів цільових атак. Це пов'язано з тим, що цей метод використовує вразливості персоналу, тобто людський фактор.

Перед атакою кіберзлочинці ретельно вивчають засоби захисту атакованої організації.

Цільова атака має наступний життєвий цикл:

– підготовка;

– проникнення;

– розповсюдження;

– виконання поставлених завдань.

Підготовка складається з вибору жертви, ошук та сбір необхідної інформації, завдяки якої можна було б виявити слабкі місця в інфраструктурі організації.

Проникнення – активна фаза цільової атаки, що проводиться для первинного інфікування мети та внутрішньої розвідки. На цьому етапі використовується цільовий фішинг.

Розповсюдження – фаза закріплення усередині інфраструктури підприємства, максимально поширюючи свій контроль.

Глибоке розслідування Cisco Talos [17] показало, що бекдор був присутній у системі принаймні шість тижнів до атаки, описуючи це як «ретельно сплановану та добре виконану операцію», що безумовно вказує на цільову атаку з використанням цільового фішингу.

Інший випадок, що також можна розцінювати як частину російсько-української кібервійни – вірус Bad Rabbit, який почав поширюватись через фішингові електроні листи. Користувачі програмного забезпечення 1С (російська програма сімейства 1С) отримували листи начебто від імені розробників цієї програми [18]. Справжньою метою атаки було отримання несанкціонованого доступу до конфіденційних та фінансових даних. Атака хробаком була лише яскравим прикриттям для відволікання уваги. І попри більше поширення хробака в Росії, друга, прихована, частина атаки служить доказом того, що основною мішенню були українські підприємства.

21 жовтня Служба безпеки України (СБУ) закликала громадян пильніше ставитися до безпеки в Інтернеті [19], насамперед – при використанні електронної пошти. СБУ зафіксувала масові фішингові розсилки нібито від її імені. Зокрема, зловмисники від імені Ситуаційного центру Департаменту кібербезпеки СБУ повідомляли про розробку спеціалізованих програм для посилення кіберзахисту та знищення вірусів. І тут же пропонували завантажити їх нібито з офіційного вебсайту СБУ. Подібні фейкові розсилки можуть бути частиною спецоперацій російських спецслужб із розхитування інформаційного та кіберпростору.

Threat Group-4127 (Fancy Bear) використовувала тактику фішингу для націлювання на облікові записи електронної пошти, пов’язані з президентською кампанією Гілларі Клінтон у 2016 році. У березні 2016 року дослідники CTU виявили кампанію підмінного фішингу з використанням облікових записів Bitly для скорочення шкідливих URL-адрес. Короткі посилання в електронних листах для підману перенаправляли жертв на URL-адресу, контрольовану TG-4127, яка підробляла законний домен Google. Рядок у кодуванні Base64, що містить повну адресу електронної пошти жертви, передається разом із цією URL-адресою, попередньо заповнюючи підроблену сторінку входу Google, яка відображається жертві. Якщо жертва вводила свої облікові дані, TG-4127 могли встановити сеанс із Google і отримати доступ до облікового запису жертви. Зловмисники могли

підтримувати цей сеанс і мати постійний доступ. Вони атакували понад 1800 облікових записів Google [20].

Дослідники CTU проаналізували 4396 фішингових URL-адрес, надісланих на 1881 обліковий запис Google у період з березня по вересень 2015 року. Більш ніж половина (59%) URL-адрес була доступна, що свідчить про те, що одержувачі принаймні відкривали фішингову сторінку. За наявними даними неможливо визначити, скільки з цих облікових записів Google було зламано. Більшість цільових облікових записів отримали кілька спроб фішингу, що може свідчити про те, що попередні спроби були невдалими. Однак 35% облікових записів, які отримали доступ за зловмисним посиланням, не були предметом додаткових спроб, що, можливо, вказує на те, що компрометація була успішною.

З початком російського вторгнення в лютому 2022 року телекомунікаційні мережі України зазнали нового удару, зокрема й з боку різноманітних фішингових атак. Так, наприклад, 25 лютого Державна служба спеціального зв'язку та захисту інформації України повідомила про масові фішингові листи на email-адреси українських сервісів i.ua та meta.ua (Рисунок 1.3). За даними Служби [22] атака була вчинена групою UNC1151, котра базується в Мінську, і до складу якої входять кадрові офіцери Міністерства оборони Республіки Білорусь.

Уважаемый пользователь.
Ваши данные не подтверждены, в связи с этим Ваш почтовый ящик в течении 2 суток будет заблокирован. Для разблокировки почтового ящика мы должны удостовериться, что Вы не являетесь спам-ботом. Для этого перейдите по ссылке ниже и подтвердите свои учётные данные: В противном случае ваш почтовый ящик будет удалён безвозвратно. Надеемся на Ваше понимание.
С уважением, команда I.UA

Рисунок 1.3 – Розміщений Службою спеціального зв'язку та захисту інформації України приклад листа

8 грудня 2022 року до Урядової команди реагування на комп'ютерні надзвичайні ситуації України CERT-UA надійшла інформація від фахівців

підрозділу кібербезпеки АТ «Укрзалізниця» про розсилку електронних листів з темою "Як розпізнати дрон-камікадзе" з адреси "morgunov.a@dsns.com[.]ua", нібито від імені Державної служби України з надзвичайних ситуацій [21]. Окрім цього, 08.11.2022 було зареєстровано відповідне доменне ім'я. До листа додавався RAR-архів "shahed-136.rar", що містив PPSX-документ "shahed.ppsx", який в свою чергу містив VBScript-код, призначений для створення запланованого завдання, а також для його розшифровки, створення на комп'ютері та запуску сценарію PowerShell. При цьому криптографічне перетворення даних виконується з використанням алгоритму RC4, а ключем є рядок, отриманий в результаті конкатенації значення властивості «Manager» та назви документа («Тригубенко Сергій Георгійович|shahed.ppsx»). Вищевказаний сценарій PowerShell використовує команду BitsTransfer (Background Intelligent Transfer Management) для завантаження виконуваних файлів "WibuCm32.dll", "CodeMeter.exe" (легітимної програми), а також для створення запланованого завдання на виконання останнього. В даному випадку використовується технологія DLL Side-Loading. Файл "WibuCm32.dll" класифікується як шкідливе програмне забезпечення DolphinCape, яке створене за допомогою мови програмування Delphi та основною функцією якого є збір інформації про комп'ютер (ім'я хоста, ім'я користувача, розрядність, версія операційної системи, змінні оточення), виконання EXE/DLL файлів, перерахування файлів та їх завантаження, а також створення та ексфільтрація скріншотів. Активність відстежується за ідентифікатором UAC-0140.

Зауважимо, що успіх фішингових атак полягає насамперед у використанні соціальної інженерії, яка спрямована на те, щоб переконати користувачів вжити негайних дій, тим самим блокуючи можливість і бажання детально аналізувати ситуацію. Тому боротьба з фішингом вимагає комплексного комплексу заходів, спрямованих на забезпечення захисту інформаційних систем та їх користувачів.

1.2 Аналіз існуючих методів виявлення фішингу

Сьогодні виявлення фішингових атак і протистояння їм – одна з найбільш досліджуваних проблем. Існує чимало робіт, пов'язаних з виявленням фішингових атак.

Один з методів виявлення фішингових сайтів базується на наявності чорного та білого списків, де зберігаються відомі URL-адреси та веб-сайти. Підозрілий сайт перевіряється за таким списком, щоб класифікувати його як фішинговий або безпечний сайт.

Метод білого списку підтримує список відомих безпечних веб-сайтів для захисту від фішингових атак, і лише ті веб-сайти, які є у списку, вважаються надійними. Загальні інструменти білого списку потребують динамічного оновлення загального білого списку, щоб забезпечити його доступність. Однак для окремих користувачів це призведе до певної надлишковості та великих витрат на обслуговування.

Хан та ін. [23], щоб вирішити цю проблему, розробили автоматизований індивідуальний білий список, в якому зберігається запис відомих безпечних сайтів, відвідуваних користувачами. Він підтримує інформацію інтерфейсу користувача, де користувач вводить свої дані, щоб запобігти нездоровому розкриттю конфіденційної інформації шкідливим сайтам. Цей метод забезпечує ефективний механізм захисту від фармінгу та динамічних фішингових атак. Проте цей метод залежить від цього, як користувачі навчають свої браузери, тобто. залежить від відгуків користувачів. Ця слабкість, як і раніше, робить користувачів, як досвідчених, так й любителів, вразливими для фішингу, якщо рівень навчання їх браузера низький. У відповідних дослідженнях Джайн і Гупта [24] розробили систему автоматичного оновлення білого списку для захисту користувачів від атак фішингу. Підхід, що складається з етапу зіставлення IP-адрес домену та етапу вилучення характеристик посилань, забезпечив швидкий час доступу та високий рівень виявлення 86,02%. Однак рівень хибно негативних

результатів у 1,48% є обмежуючим фактором у критичній системі онлайн-транзакцій.

Метод чорного списку в основному реалізує ідентифікацію фішингових атак шляхом ведення списку відомих фішингових сайтів і перевірки відвідуваних веб-сайтів. Чорні списки зазвичай складаються зі списків шкідливих веб-сайтів, зібраних з кількох джерел даних, таких як фільтри спаму, повідомлення користувачів або перевірені фішингові сайти, складені третіми сторонами. Недоліком цього підходу є те, що він не ідентифікує всі фішингові веб-сайти. Оскільки після того, як фішинговий сайт закрито, фішер може легко зареєструвати новий домен.

Фішингову ідентифікацію на основі списків легко реалізувати, вона має таку перевагу, як висока швидкість роботи. Однак, оскільки невелика різниця між URL-адресою, яку потрібно виявити, і списком призведе до помилки відповідності, зловмисники часто уникають виявлення, змінюючи URL-адресу фішингової веб-сторінки. У той же час статистика показує [25], що 63% фішингових веб-сайтів мають життєвий цикл лише 2 години, але від 47 до 83% фішингових веб-сайтів будуть додані до чорного списку через 12 годин. Через високу швидкість генерації та короткий життєвий цикл фішингових веб-сайтів чорний список потрібно часто оновлювати з джерела, але це призведе до значного використання системних ресурсів, а система на основі списку не зможе виявити фішингові атаки нульового дня.

Рао, Вайшнаві та Паіс запропонували метод для класифікації за допомогою класифікатора випадкового лісу [26]. Він кодує інформацію URL-адреси у двовимірний тензор і передає тензор у нейронну мережу глибокого навчання для класифікації вихідної URL-адреси. Вони використовують спочатку мережу двонаправленої довгострокової пам’яті (LSTM), а потім згортову нейронну мережу (CNN) для вилучення глобальних і локальних характеристик URL-адреси.

Юань та ін. [27] розробили метод, який поєднує вбудовування символів зі структурами URL-адрес для отримання векторних представлень URL-адрес. Вони

поділяють URL-адресу на п’ять частин: протокол URL-адреси, ім’я піддомену, ім’я домену, суфікс домену та шлях URL-адреси. Векторне представлення URL-адрес навчено існуючими алгоритмами класифікації для ідентифікації фішингових URL-адрес.

Хунг Ле та ін. [28] запропонували математичну модель URLNet (Рисунок 1.4) – глибоку нейронну мережу на основі CNN для виявлення шкідливих URL-адрес (Рисунок). У своїй роботі вони застосовують згорточні нейронні мережі як до символів, так і до слів рядка URL-адреси, щоб дізнатися про вбудовування URL-адреси в спільно оптимізовану структуру. Такий підхід дозволяє моделі фіксувати кілька типів семантичної інформації.

URL – це, по суті, послідовність символів або слів (розмежованих спеціальними символами). Автори URLNet прагнули отримати його матричне представлення $u \to x \in R^{Lxk}$ таким чином, щоб екземпляр x містив набір суміжних компонентів $x_i, i = 1, \cdots, L$ у послідовності, де компонентом може бути символ або слово URL-адреси . Кожен такий компонент представлений вкладенням таким, що $x_i \in R^k$, є k-вимірним вектором.

Зазвичай це двовимірне представлення для компонента є вектором вбудовування, витягнутим із матриці вбудовування, яка випадковим чином ініціалізується та вивчається спільно з рештою моделі. У свої роботі Хунг Ле випадковим чином ініціалізував матрицю вбудовування та вивчав її під час наскрізної оптимізації. За допомогою цієї нотації екземпляр із послідовністю L компонентів можна представити як:

$$x = x_{1:L} = x_1 \oplus x_2 \oplus \ldots \oplus x_L \tag{2.1}$$

де $\oplus$ — оператор конкатенації. З метою розпаралелювання зазвичай усі послідовності доповнюються або скорочуються до однакової довжини L.

CNN буде згортатися над цим екземпляром $x \in R^{L*k}$ за допомогою оператора згортки. Операція згортки $\otimes$ довжиною h складається зі згортання фільтра $W \in R^{k*k}$ з наступною нелінійною активацією f для отримання нової функції:

$$c_i = f(W \otimes x_{i:i+h-1} + b_i) \quad (2.2)$$

де b_i — зміщення. Вихід цього шару згортки застосовує фільтр W з нелінійною активацією до кожного сегмента h-довжини його входу, кожен з яких розділений заздалегідь визначеним значенням кроку. Потім ці виходи об'єднуються для отримання виходу с таким чином, що:

$$C = [C_1, C_2, \dots, C_{L-h+1}] \quad (2.3)$$

Після згортки застосовується крок об'єднання (або максимальне, або середнє об'єднання), щоб зменшити розмірність функції та визначити найважливіші функції.

Використовуючи фільтр W для згортання на кожному сегменті довжини h, CNN може використовувати тимчасове співвідношення довжини h у своїх вхідних даних. Модель CNN зазвичай складається з кількох наборів фільтрів різної довжини (h), і кожен набір складається з кількох фільтрів. Це гіперпараметри моделі, які повинен задати користувач. Згортка, за якою слідує шар об'єднання, становить блок у цій глибокій нейронній мережі. Таких блоків може бути кілька, які можна складати один на одного. Об'єднані об'єкти з останнього блоку об'єднуються та передаються на повністю пов'язані рівні з метою класифікації. Потім мережу можна навчити шляхом стохастичного градієнтного спуску з використанням зворотного поширення.

URLNet використовує кілька CNN: один для рівня символів і один для рівня слів.

Метод розпізнавання, заснований на машинному навчанні, може ідентифікувати фішингові атаки нульового дня та має такі переваги, як висока точність і масштабованість, а також відносно проста корекція моделі (достатньо додати нові фішингові дані до вихідного набору даних). Але продуктивність його розпізнавання тісно пов'язана з розподілом вибірок даних. Якщо розповсюдження зразків фішингу та законних веб-сайтів суттєво відрізняється, це значно вплине на ефективність розпізнавання. У той же час через широку різноманітність і масштабність функцій, які використовуються для ідентифікації фішингу,

наявність надлишкових функцій часто призводить до збільшення вартості технології (зберігання, час навчання тощо).

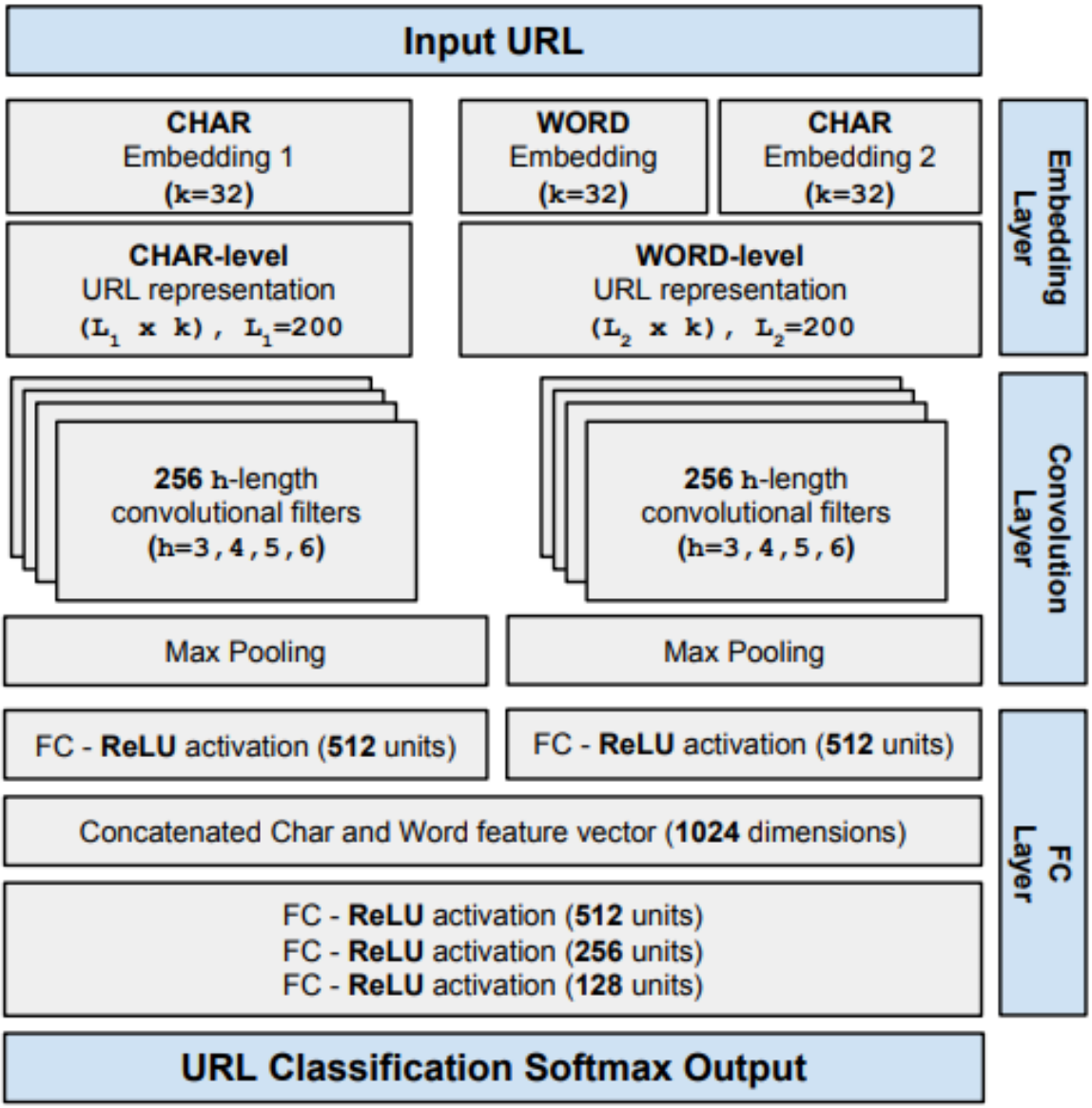


Рисунок 1.4 – Модель URLNet

Уго Гаскон та ін. у своїй статті [29] показали, що відправник залишає в структурі електронного листа риси, що не залежать від вмісту. На основі цих характеристик вони розробили метод, здатний вивчати профілі для великої групи відправників і ідентифікувати підроблені електронні листи як відхилення від них. Недоліком цього методу є низька продуктивність у випадку, якщо зловмисник має

доступ до електроних листів, що надходять з того самого домену. У найгіршому випадку зловмисник має достатньо інформації, щоб створити електронний лист, який нагадує вивчений профіль підробленого відправника, що спричиняє значне погіршення продуктивності класифікатора.

Орунсолу та ін. запропонували прогностичну модель для виявлення фішингу [30].

Визначення проблеми фішингової атаки є типовим випадком проблеми двійкової класифікації, оскільки онлайн-комунікація (наприклад, веб-сайт, електронна пошта чи електронний чат) є фішинговою або доброякісною. Більш формально, нехай w буде запитом, який потребує класифікації, тобто

$$w \xrightarrow{x} \{phish, benign\}, \tag{2.4}$$

тоді X — це система захисту від фішингу, яка має функції $f_i \in w$.

$$w = \sum_i^n f_i n > 0 \tag{2.5}$$

Таким чином, запит містить принаймні одну функцію (наприклад, посилання, HTML-теги, сценарії, SSL-сертифікат тощо), за якими можна запитувати чи класифікувати передбачення його статусу. Оскільки ці ознаки можуть варіюватися від простих до складних, запропонована модель використовує оцінку частоти ознак для векторної композиції ознак, зображеної $x = \{x_1 x_2 \dots x_n\}$, які присвоюють мітку y кожному $f_i \in w$, так що мітка y є бінарним класом, представленим у вигляді:

$$y = \begin{cases} 1\ (phishing), \\ 0\ (other) \end{cases} \tag{2.6}$$

Представлений як

$$x_i : f(w) \to y \tag{2.7}$$

Рівняння 2.4 описує проблему класифікації, де, задані навчальні дані *D*, які містять $\{w_1 w_2 \dots w_n\}$. І кожен w_i містить набір функцій $\{f_1 f_2 \dots f_n\}$. Крім того, навчальні дані - це набір класів $C = (C_1, C_2)$ який представляє фішингові та законні сайти, такі як:

$$C_1 = \{w_i, f_i | w_i \in D, y = phishing, i = 1 \dots m\} \quad (2.8)$$

$$C_2 = \{w_i, f_i | w_i \in d, y = phishing, i = m + 1, \dots p\} \quad (2.9)$$

Таким чином, кожен випадок $w_i \in D$ може бути наданий клас $c_i \in C$ і представляється як пара $(w_i, (c_i))$ де c_i клас з C, пов'язаний з регістром у даних навчання. Нехай H позначає набір класифікаторів для $D \to C$, де кожен випадок $c_i \in C$ дається клас, а мета — знайти класифікатор $h_i \in H$, що максимізує ймовірність того, що $h(ci) = c$ для кожного тесту. У запропонованій моделі два найпоширеніші класифікатори машинного навчання для класифікації фішингу, а саме Naive Bayes і Support Vector Machine, вибрано для дослідження продуктивності набору функцій/вектора та максимізації точності запропонованого підходу.

Недоліком запропонованої моделі виявлення фішингу є той випадок, якщо фішер імітує веб-сторінку за допомогою складних інструментів, тоді зовнішній вигляд такого веб-сайту може бути копією законної сторінки. У цьому випадку використання візуальних атрибутів може не дати істотних позитивних результатів.

Таким чином, питання протидії фішинговим атакам в Україні та у всьому світі загалом залишається дуже гострим. Користувачі Інтернету постійно стикаються зі спробами заволодіння їх особистими чи фінансовими даними. Жертвами можуть стати як звичайні користувачі, так і великі компанії, що, безумовно, може поставити під загрозу безпеку держави, якщо фішингова атака є частиною кібервійни.

Отже, не зважаючи на різні підходи до вирішення проблеми фішинга, він залишається гострою проблемою кібербезпеки. Проблема полягає у тому, що існуючі методи забезпечення безпеки інформаційних систем від фішингових атак у більшості не акцентують увагу на способі розповсюдження фішингу, що значно би полегшило створення оптимального захисту.

2. ОПИС АЛГОРИТМА ВИЯВЛЕННЯ ФІШИНГОВИХ ЕЛЕКТРОННИХ ЛИСТІВ

2.1 Аналіз заголовків електронних листів

Зазвичай, фішинг можна виявити за допомогою методів виявлення на основі списків, машинного навчання, евристичного виявлення або глибокого навчання. Однак проблема фішингу настільки складна, що немає вирішального рішення, яке б ефективно подолало всі загрози; отже, для запобігання конкретним атакам часто використовуються кілька методів.

Заголовок листа – це частина коду, яка містить важливі дані для автентифікації електронного повідомлення. Він передує тексту листа, а також містить інформацію про відправника та отримувача.

До заголовків відносяться не лише розділи «кому», «від кого», «дата» та «тема», які відображаються користувачеві при отриманні листа. Заголовок також відіграє важливу роль у фіксації маршруту електронного листа, оскільки кожне повідомлення електронної пошти має заголовок.

Коли електронний лист надсилається з однієї адреси на іншу, він проходить через поштових агентів (MTA). Таким чином, у заголовку електронного листа буде видно: чи надсилався лист на інші адреси до того, як він потрапив до кінцевого одержувача; інформацію про те, чи схожий цей лист на спам; інформацію про перевірку на віруси; рівень терміновості листа тощо.

Враховуючи певний відсоток листів, які оминули фільтрацію на стороні поштового клієнта, запропонований алгоритм містить аналіз заголовків електронних листів.

Підробка електронної пошти – техніка, яка використовується під час атак зі спамом і фішингом, щоб змусити користувачів подумати, що повідомлення надійшло від особи чи організації, яку вони знають або якій можуть довіряти. Під час спуфінгу відправник підробляє заголовки електронної пошти, щоб клієнтське програмне забезпечення відображало шахрайську адресу відправника, яку більшість користувачів сприймають за чисту монету. Якщо вони не перевірять заголовок уважніше, користувачі побачать у повідомленні підробленого відправника. Якщо це ім'я, яке вони впізнають, вони, швидше за все, довірятимуть йому. Тож вони натискатимуть шкідливі посилання, відкриватимуть вкладені файли зловмисного програмного забезпечення , надсилатимуть конфіденційні дані та навіть переведуть корпоративні кошти.

Як приклад спуфінгу електронної пошти, зловмисник може створити електронний лист, який виглядає так, ніби він надійшов від PayPal. У повідомленні користувачеві повідомляється, що його обліковий запис буде призупинено, якщо він не натисне посилання, не пройде автентифікацію на сайті та не змінить пароль облікового запису. Якщо користувача вдалось обдурити та він вводить облікові дані, зловмисник тепер має облікові дані для автентифікації в обліковому записі цільового користувача PayPal, що потенційно може вкрасти гроші користувача.

Більш складні атаки націлені на фінансових працівників і використовують соціальну інженерію та онлайн-розвідку, щоб обманом змусити цільового користувача надіслати мільйони на банківський рахунок зловмисника.

Як показує практика, люди майже не переглядають заголовки отриманих листів, та, не маючи необхідних знань, не можуть провести мінімальний аналіз заголовків.

У роботі пропонується проведення аналізу наступних частин заголовків електронного листа для запобігання фішингу:

– Return-Path;
– Reply-To;
– Received-SPF;
– DKIM (Domain Keys Identified Mail).

Адреса електронної пошти Return-Path містить інформацію про статус відправлення. Поштовий сервер зчитує вміст заголовка Return-Path для обробки повідомлень, які не можна доставити або повернути відправнику. Сервер-одержувач використовує це поле для ідентифікації «підроблених» електронних листів: він запитує всі дозволені IP-адреси, пов'язані з доменом відправника, і порівнює їх з IP-адресою автора повідомлення. Якщо вони не знайдуть збігів, лист відправляється в спам.

На рисунку 2.1 представлена блок-схема перевірки Return-Path, на виході якої отримуємо n_1 – бали фішингу за рівність *From = Return-Path*.

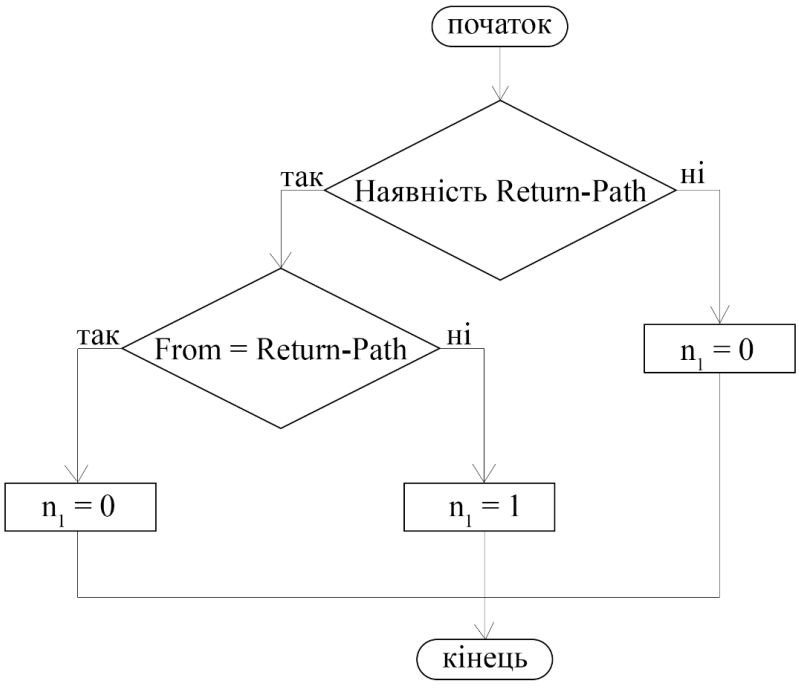


Рисунок 2.1 – Блок-схема перевірки Return-Path

Спочатку перевіряється наявність заголовка Return-Path. У деяких випадках, наприклад, при автоматичному повідомлені від системи безпеки, цей заголовок може бути відсутній. Якщо поле заголовка знайдено, воно зіставляється з полем заголовка From. При співпадінні *n1 = 0,* якщо збігів не знайдено, то $n_1 = 1$.

Адреса електронної пошти в полі Reply-To використовується для надсилання відповіді. У фальшивих електронних листах вона може відрізнятися від адреси відправника. На рисунку 2.2 представлена блок-схема перевірки Reply-To, на виході якої отримуємо n_2 – бали фішингу за рівність *From = Reply-To*.

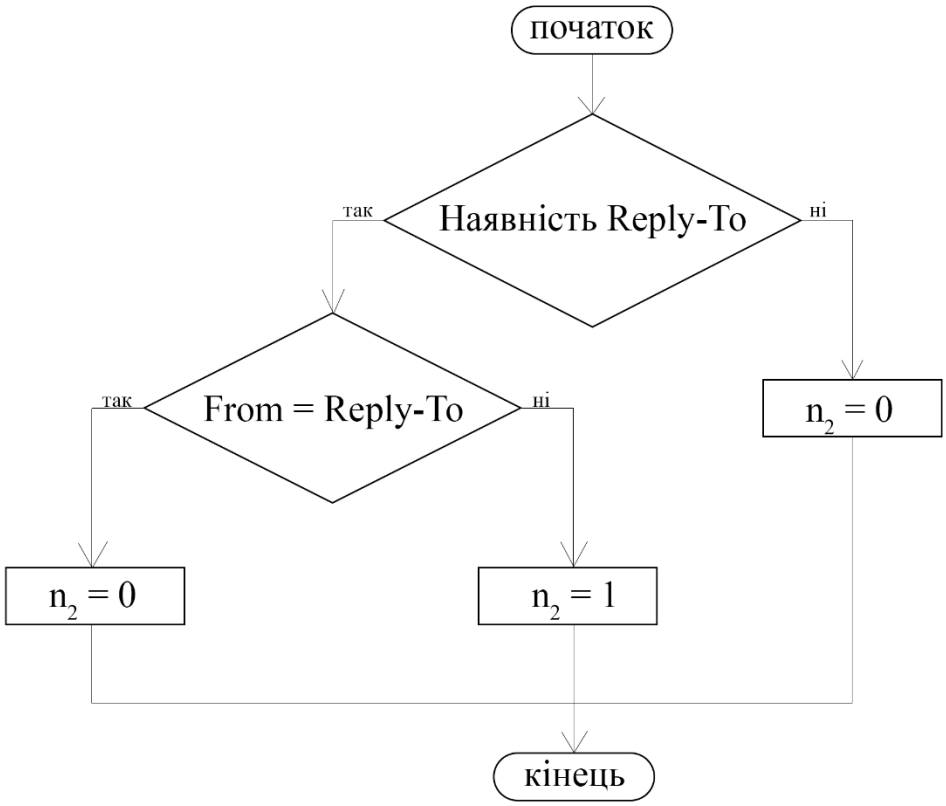


Рисунок 2.2 – Блок-схема перевірки Reply-To

Як і в випадку Return-Path, спочатку перевіряється наявність заголовка Reply-To. Якщо поле заголовка знайдено, воно зіставляється з полем заголовка From. При співпадінні $n_2 = 0$, якщо збігів не знайдено, то $n_2 = 1$.

Received-SPF дозволяє одержувачу перевірити, що електронна пошта, яка нібито походить з певного домену, походить з IP-адреси, дозволеної адміністраторами цього домену. Адміністратор домену зазвичай авторизує IP-адреси, які використовують його власні вихідні MTA, включаючи будь-які проксі-сервери або смарт-хости [31].

При встановленні з'єднання протокол управління трафіком перевіряє IP-адресу відправника MTA і переконується, що віддалений хост доступний. Поштовий сервер-одержувач отримує команду HELO SMTP незабаром після встановлення з'єднання і команду Mail from: на початку кожного повідомлення. Обидва можуть включати в себе доменне ім'я. Верифікатор SPF запитує в системі доменних імен (DNS) відповідний SPF-запис, який, за наявності, ідентифікує IP-адреси, дозволені адміністратором домену. Це поле є дійсним, якщо воно має значення PASS.

Блок-схема перевірки заголовка Received-SPF представлена на Рисунку 2.3, на виході блок-схеми отримуємо n_3 – бали фішингу за рівність *Received-SPF = PASS*.

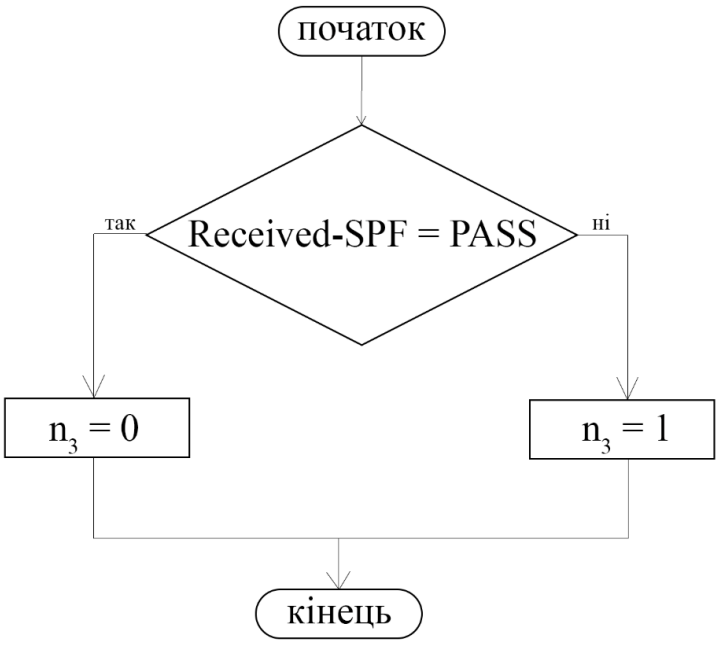


Рисунок 2.3 – Блок-схема перевірки Received-SPF

При співпадінні $n_3 = 0$, якщо збігів не знайдено, то $n_3 = 1$.

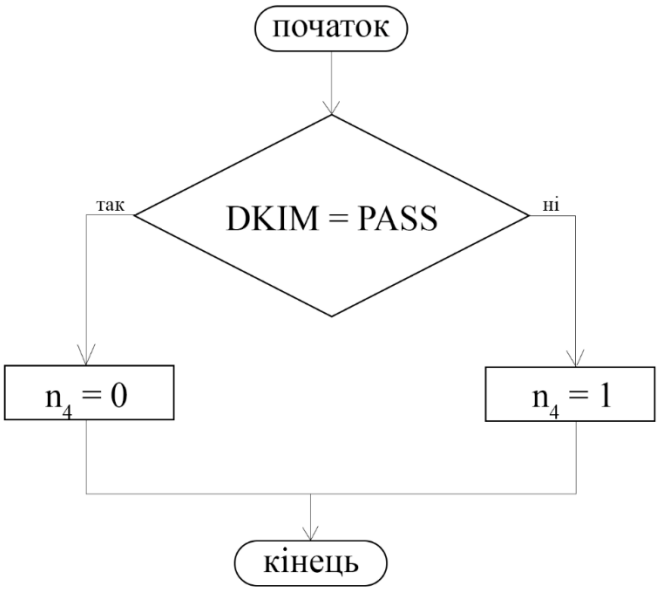
DKIM перевіряє зміст повідомлень за допомогою цифрових підписів. Замість цифрових сертифікатів через DNS розповсюджуються ключі перевірки підпису. Таким чином, повідомлення асоціюється з доменним ім'ям.

Адміністратор DKIM-сумісного домену створює одну або більше пар асиметричних алгоритмів шифрування, потім відправляє закриті ключі підписуючим MTA і публікує відкриті ключі в DNS. DNS мітки структуровані у вигляді selector._domainkey.example.com, де селектор вказує пару ключів, а _domainkey – фіксоване ключове слово, за яким слідує назва підписаного домену, щоб публікація перебувала під контролем ADMD цього домену. Безпосередньо перед тим, як вставити повідомлення в транспортну систему SMTP, підписуючий MTA створює цифровий підпис, який охоплює вибрані поля заголовка і тіла (або тільки початок). Підпис повинен включати основні поля заголовка, такі як «Від кого», «Кому», «Дата» і «Тема», а потім додаватися до самого заголовка повідомлення як поле для відстеження.

Повідомлення можуть прийматися і відправлятися будь-якою кількістю ретрансляторів, і на кожному кроці підпис може бути перевірений шляхом отримання відкритого ключа з DNS [32]. Поки посередники не змінюють підписані частини повідомлення, підписи повідомлень DKIM залишаються дійсними.

Це поле є дійсним, якщо воно має значення PASS.

Блок-схема перевірки заголовка DKIM представлена на рисунку 2.4.



,

Рисунок 2.4 – Блок-схема перевірки DKIM

На виході блок-схеми отримуємо n_4 – бали фішингу за рівність *DKIM = PASS*. При співпадінні $n_4 = 0$, якщо збігів не знайдено, то *n4 = 1*.

2.2 Семантичний аналіз вмісту електронного листа

Для аналізу вмісту листа у роботі використовується семантичний аналіз для порівняння слів та словосполучень. Порівняння здійснюється за допомогою двох словників: спаму та тональності (настроїв), аналізуючи слова, вирази та символи, як видимі, так і приховані для людського ока, за допомогою різних методів.

У роботі використовується словник OOPSpam та український тональний словник [33], який містить 3442 слів української мови, які мають не нейтральну тональність.

Словник спаму містить у собі слова та словосполучення з типовою лексикою спаму. На Рисунок 2.5 зображена блок-схема перевірки листа за словником спаму, на виході якої отримуємо n_6 – кількість балів за перевірку листа за словником спаму. Кожне слово листа зіставляється зі словником, при виявленні збігів (w_2) відбувається перевірка умов. Якщо збігів менше двох – $n_6=0$, якщо у діапазоні між двома та п'ятьома – $n_6=1$, якщо більше п'яти, то $n_6=2$.

Мета аналізу настроїв полягає в аналізі певної кількості даних, щоб визначити різні почуття, виражених в них. Отримані почуття потім можуть бути предметом статистичних даних щодо загального відчуття спільноти. Прикладом використання тональності тексту є робота Мангурі та ін. [34], які проводили аналіз настроїв записів користувачів у Twitter, щодо спалахів COVID-19 у всьому світі.

Сентимент-аналіз дозволяє аналізувати листи «вимагачі», «жебраки» або «благодійні», які можуть бути надіслані з реальних адрес та не містити жодного посилання.

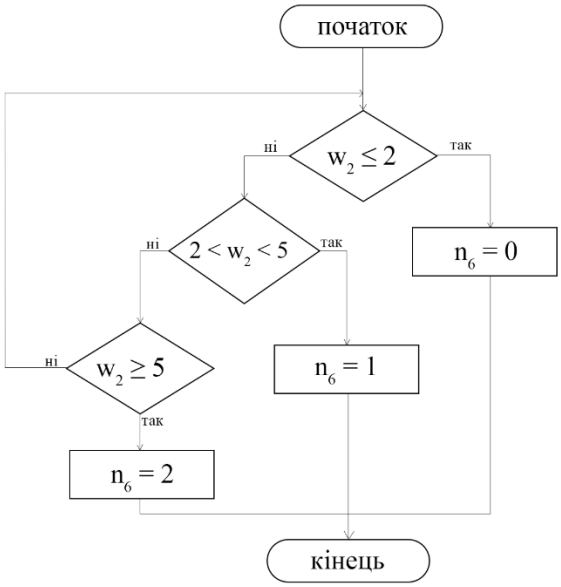


Рисунок 2.5 – Блок-схема перевірки листа за словником спаму

У цьому разі фішери використовують як зброю слова, щоб викликати у жертви негативні або позитивні відчуття. Зазвичай виділяють два види емоцій: позитивні та негативні, тобто класифікація виконується за двома класами.

Існує дві великі категорії аналізу: лексичний аналіз і аналіз з використанням машинного навчання. У роботі використовується лексичний аналіз.

Підхід, заснований на лексичному аналізі, полягає у виведенні емоцій, які викликає речення, за допомогою семантичного аналізу слів. Цей підхід включає класифікацію речення за допомогою вже існуючих екземплярів речень, для яких емоції вже ідентифіковано. Для цього був використаний словник , який посилається на слова.

Кожне слово тексту зіставляється зі словником, якщо знайдене співпадіння, то тональність тексту зростає. На рис.2.6 зображена блок-схема перевірки листа за словником тональності, на виході якої отримуємо n_5 – кількість балів за перевірку листа за словником тональності.

Кожне слово листа зіставляється зі словником, при виявленні збігів (w_1) відбувається перевірка умов. Якщо збігів менше двох – $n_5=0$, якщо у діапазоні між двома та п’ятьома – $n5=1$, якщо більше п’яти, то $n5=2$.

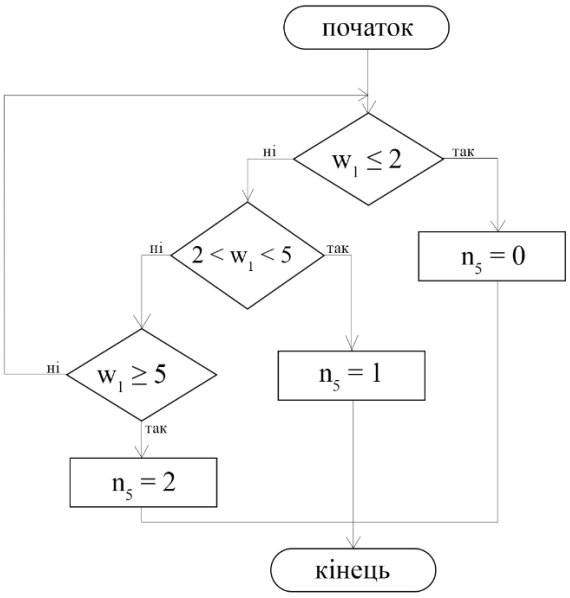


Рисунок 2.6 – Блок-схема перевірки листа за словником тональності

У обох випадках якщо кількість збігів w_n (n – номер словника) більше за п’ять, то бали фішинга дорівнюють двом. Зловмисники мають на меті приховати свою особистість та за допомогою маніпулятивних дій – речень – завдати користувачеві матеріальної або психологічної шкоди. У роботі припускається, що зловмисники можуть використовувати тактики маркетингу для збільшення імовірності ввести потенційну жертву в оману. Однією з таких тактик є метод «холодного листа». Холодний лист – це перший лист, який надсилається потенційному клієнту, тобто перший контакт з людиною.

Таблиця 2.1 – Відсоток збігів w_n

Кількість слів	Відсоток збігів, %
50	10
75	6,7
100	5
125	4
150	3,3
175	2,8
200	2,5

Дослідження [35-36] показали, що ідеальна довжина для листа (щоб він міг вважатися «холодним») варіюється від 50 до 200 слів. На основі цих значень було припущено, що кількість слів-збігів не може перевищувати за 10% від загальної кількості слів листа (таблиця 2.1) та обрано середнє значення з можливих.

2.3 Аналіз наявних URL-посилань у листі

Зміст листа також може містити посилання. У цьому випадку метою зловмисника є перенаправлення жертви на певний веб-ресурс, куди зловмисник намагається впровадити шкідливе програмне забезпечення, використовуючи вразливості в самій сторінці або при переході в браузері.

URL-фішинг – це шахрайська практика заманювання людей на фальшиві сайти з переконливим змістом, де вони завантажують шкідливе програмне забезпечення або розкривають конфіденційну інформацію, таку як імена користувачів, паролі та банківські реквізити. Один з найпоширеніших прикладів такої атаки – коли шахраї імітують компанію та надсилають електронного листа з наступним повідомленням: «Ваш обліковий запис заблоковано. Відновити його можна за посиланням». Ошуканий користувач переходить за посиланням і несвідомо завантажує шкідливе програмне забезпечення або переходить на підроблений веб-сайт, який виглядає легітимно.

Після переходу за посиланням користувач може стати жертвою різного роду XSS-атак. Суть цих атак полягає у виконанні скрипту в браузері та його подальшій взаємодії з сервером зловмисника. Ці операції відкривають доступ до даних браузера і дозволяють ввести в нього експлойтів, а також викрадати файли cookies, дані авторизації або, наприклад, здійснювати HTTP-запити від імені користувача. За даними Звіту про розслідування порушень даних [37] близько 91% порушень безпеки починаються з фішингової атаки, і багато з них включають шкідливі посиланням на підроблені сайти.

У роботі URL-адреса аналізується за наступними критеріями:

– наявність символу «@»;
– довжина URL-адреси;
– кількість скісних рисок.

Зловмисники часто використовують подвійні скісні риски, щоб приховати шахрайську частину URL-адреси. Якщо URL-адреса містить забагато символів «/», вона є фішинговою, в іншому випадку – легітимною.

На Рисунок 2.7 зображена блок-схема перевірки URL-посилань на кількість символів «/» – *slash*, на виході якої отримуємо n_7 – кількість балів фішингу. Якщо *slash* більше або дорівнює п’яти, n_7 дорівнює одиниці, в іншому випадку – нулю.

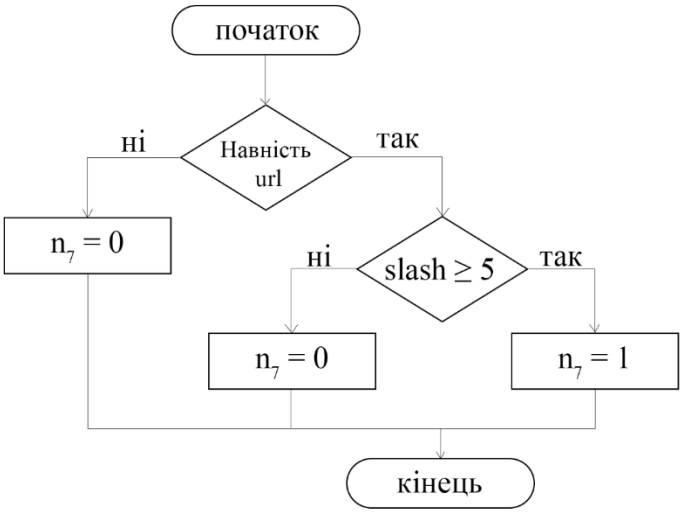


Рисунок 2.7 – Блок-схема перевірки URL-посилань на кількість символів «/»

Підрахунок кількості символів в URL-адресі є важливою характеристикою для виявлення шахрайських джерел. Зловмисники використовують довгі URL-адреси, щоб приховати шахрайську частину адреси в адресному рядку. Таким чином, видима частина адресного рядка містить легітимну URL-адресу, яка може вводити в оману.

Якщо довжина URL-адреси перевищує 35-символьний ліміт, джерело вважається підозрілим, в іншому випадку джерело легітимне[38].

На рисунок 2.8 зображена блок-схема перевірки URL-посилань на загальну кількість символів (*len*), на виході якої отримуємо n_8 – кількість балів фішингу. Якщо *len* більше або дорівнює тридцяти п’яти, n_7 дорівнює одиниці, в іншому випадку – нулю.

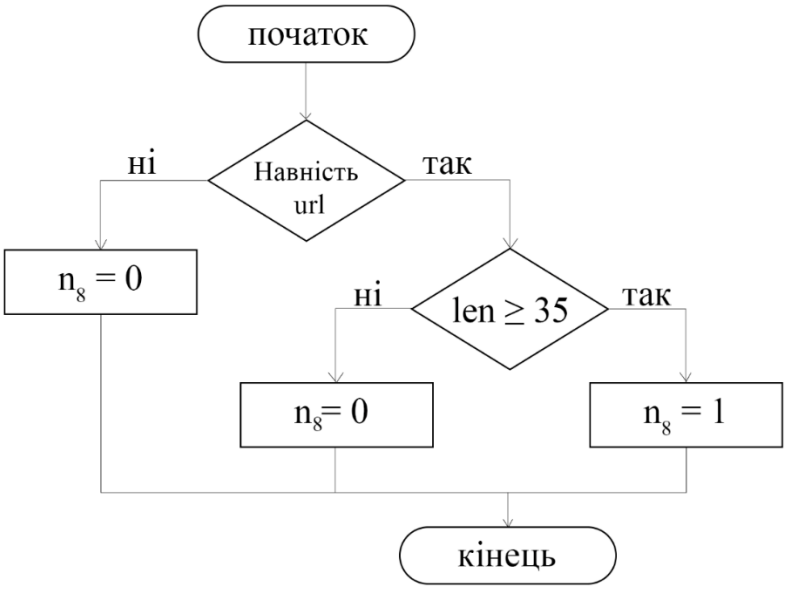


Рисунок 2.8 – Блок-схема перевірки URL-посилань на загальну кількість символів

Символ равлика «@» використовується для перенаправлення трафіку на інший, як правило шахрайський, сайт доменне ім'я якого одразу супроводжується символом @. Наприклад, http://op.edu.ua@download.file.com перенаправить користувача на download.file.com замість op.edu.ua – усе, що стоїть перед равликом відкидається. Як правило, цей синтаксис сьогодні практично не використовується. Так, якщо в адресному рядку з'являється символ «@», то URL-адреса вважається підозрілою, в іншому випадку – легальною.

Блок-схема на рисунку 2.9 перевіряє наявність в URL символу равлика, та, якщо він є, кількість балів за дану перевірку (n_9) дорівнюватиме одиниці, в іншому випадку – нулю.

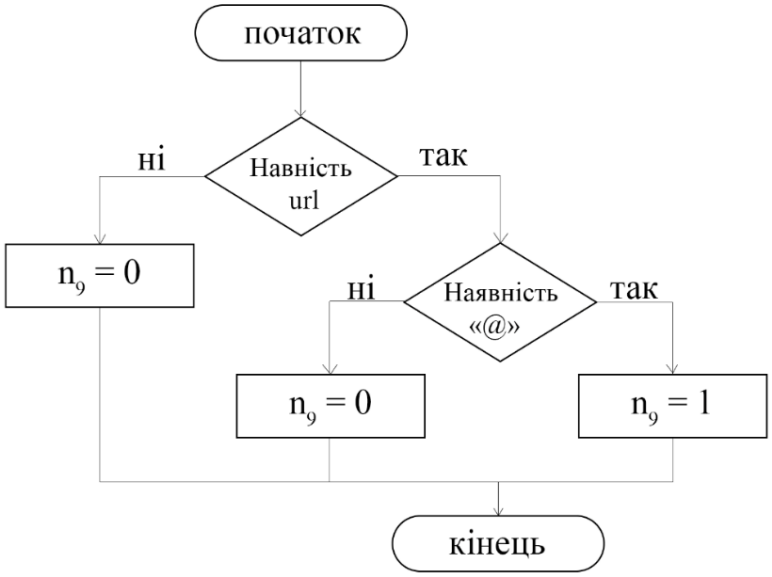


Рисунок 2.9 – Блок-схема перевірки URL-посилань на наявність символу равлика

Загальне представлення блок-схеми алгоритму представлено у додатку А.

У таблиці 2.2 відображена максимальна кількість балів, яку може отримати кожний критерій аналізу окремо.

Таблиця 2.2 – Розподіл балів фішингу

Аналіз								
Заголовки				Вміст листа				
Return-Path (n_1)	Reply-To (n_2)	Received-SPF (n_3)	DKIM (Domain Keys Identified Mail (n_4)	Словник тональності тексту (n_5)	Словник «спаму» (n_6)	Наявність посилання		
						Символ @ (n_7)	Кількість // (n_8)	Довжина url (n_9)
1	1	1	1	2	2	1	1	1

Після отримання балів, вони підсумуються:

$$points = n_1 + n_2 + n_3 + n_4 + n_5 + n_6 + n_7 + n_8 + n_9, \quad (3.1)$$

де *points* – загальна кількість балів.

Небезпечність листа залежить від кількості балів (таблиця 2.3).

Таблиця 2.3 – Значення загальної кількості балів

Результат	Кількість балів (points)
Небезпечно	points > 3
Сумнівно	2 ≤ points ≤ 3
Безпечно	points < 2

Розподіл балів розроблений таким чином допомагає уникнути хибних результатів. Так, наприклад, наявність лише спам-слів або слів з вираженим емоційним забарвленням, не зробить лист стовідсотково фішинговим.

Отже, у цьому розділі були розглянуті основні критерії оцінювання ступеня небезпеки електронного листа та поетапно описаний запропонований алгоритм аналізу.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМА ВИЯВЛЕННЯ ФІШИНГОВИХ ЕЛЕКТРОННИХ ЛИСТІВ

3.1 Загальні відомості про веб-браузер

Google Chrome – це кросплатформний веб-браузер, розроблений Google. Вперше він був випущений як бета-версія 2 вересня 2008 році для Microsoft Windows, створений за допомогою безкоштовних програмних компонентів Apple WebKit і Mozilla Firefox [39]. «Стабільна» публічна версія була випущена 11 грудня 2008 року з підтримкою 43 мов.

У якості програмної реалізації запропонованого гібридного методу аналізу, було розроблено розширення для браузера Athena logic, щоб дозволити користувачеві класифікувати електронні листи у своєму клієнті веб-пошти як легітимні або фішингові.

Як показано на рисунку 3.1 станом на жовтень 2022 Google Chrome має 66,7% користувачів по світу [40] (після піку в 72,38% у листопаді 2018 року) на персональних комп’ютерах.

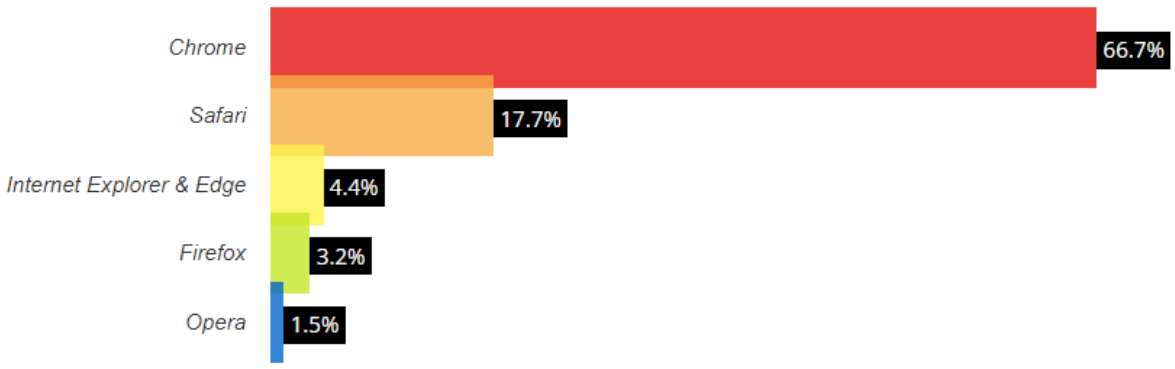


Рисунок 3.1 – Статистика популярності браузерів у світі

Та близько 60% користувачів в Україні [41] (Рисунок 3.2).

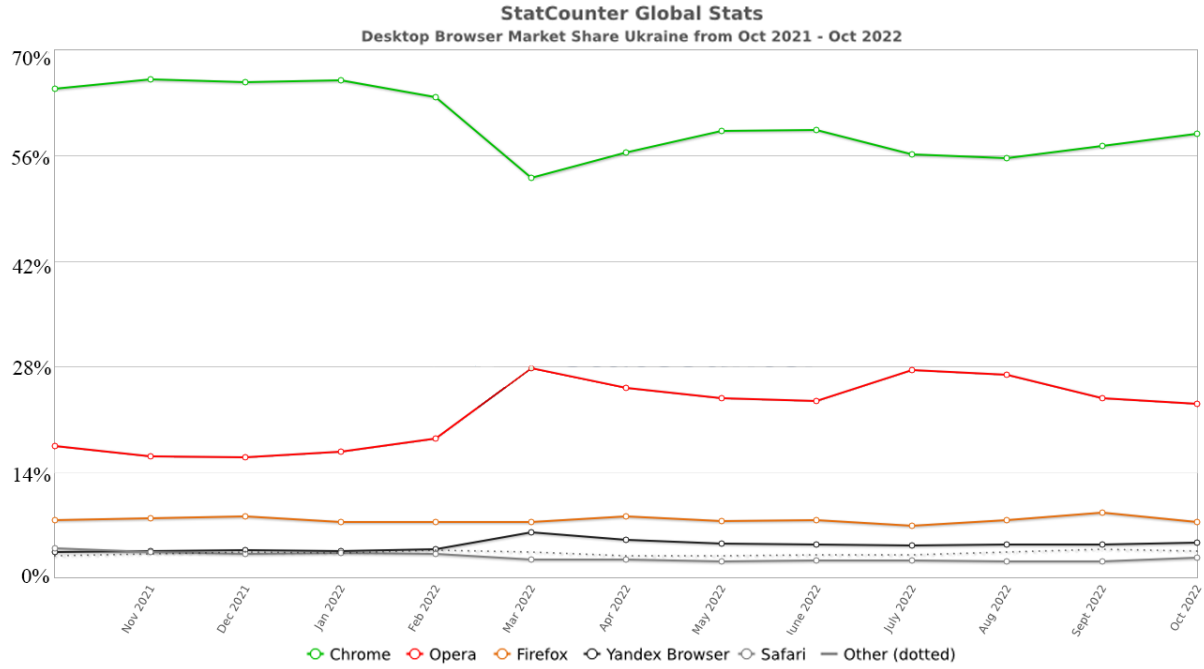


Рисунок 3.2 – Статистика популярності браузерів на ПК в Україні

Саме тому розширення для веб-браузера було розроблено для Chrome, бо він залишається доступним для найбільшої кількості користувачів.

Athena logic було розроблене згідно до документації Google developer [42].

Розширення браузера можуть змінювати Google Chrome. Вони підтримуються настільною версією браузера [43]. Ці розширення написані з використанням таких веб-технологій, як HTML, JavaScript і CSS [44]. Вони поширюються через веб-магазин Chrome (раніше – Google Chrome Extensions Gallery).

9 вересня 2009 року Google за замовчуванням увімкнув розширення на каналі розробників Chrome і надав кілька зразків розширень для тестування [45]. У грудні почалася бета-версія Google Chrome Extensions Gallery з приблизно 300 розширеннями [43]. Він був запущений 25 січня 2010 року разом із Google Chrome 4.0, що містить приблизно 1500 розширень.

Користувачі Інтернету можуть використовувати розширення браузера, щоб оптимізувати своє використання браузеру. Розширення веб-браузера –

це доповнення до обраного користувачем веб-переглядача, які можуть вносити зміни на стороні клієнта, щоб змінити спосіб відображення веб-сторінок або надавати користувачеві інструменти, які допомагають йому під час перегляду Інтернету.

Розширення браузера можна використовувати для виявлення різних форм фішингових атак, наприклад, «GoldPhish» — це розширення Internet Explorer, яке використовується для виявлення фішингових веб-сторінок [46посилання на приклад].

3.2 Опис використаних мов програмування та засобів для програмної реалізації

Розширення було розроблене завдяки інтерпретованій системно-незалежній об’єктно-орієнтованій мові програмування JavaScript, мові тегів HTML, засобами якої здійснюється розмічання веб-сторінок для мережі Інтернет, та спеціальної мови стилю сторінок CSS, що використовується для опису їхнього зовнішнього вигляду.

Мова JavaScript використовується для:

– написання сценаріїв вебсторінок для надання їм інтерактивності;

– створення односторінкових та прогресивних вебзастосунків (React, AngularJS, Vue.js);

– програмування на боці сервера (Node.js);

– стаціонарних застосунків (Electron, NW.js);

– мобільних застосунків (React Native, Cordova);

– сценаріїв в прикладних програмах;

– всередині PDF-документів тощо.

Для написання розширення було використано інтегроване середовище розробки для кодування на JavaScript і пов’язаних з ним технологіях –

WebStorm від чеської компанію розробки програмного забезпечення JetBrains.

Деякі ключові функції WebStorm [47]:

– Готова підтримка JavaScript, TypeScript, React, Vue, Angular, Node.js, HTML, таблиць стилів та інших.

– Розумний редактор із доповненням коду , миттєвим виявленням помилок , безпечним рефакторингом коду та швидкою навігацією по всій кодовій базі .

– Різноманітність вбудованих інструментів розробника , які дозволяють налагоджувати та тестувати програми на стороні клієнта та Node.js, а також працювати з керуванням версіями , лінтерами , інструментами збірки , терміналами та клієнтами HTTP .

– Інструменти для ефективної командної роботи , включаючи сервіс для віддаленої спільної розробки та парного програмування , а також можливість ділитися конфігурацією проекту з іншими.

– Можливість налаштовувати робоче середовище, експериментуючи з такими речами, як теми та плагіни .

3.3 Інтерфейс розширення та його архітектура

Розширення використовує зелені та червоні кольори, щоб підкреслити, що користувач повинен і не повинен робити у відповідь на отримання потенційного фішингового електронного листа. В універсальних кольорових кодах, таких як ті, що слідують за світлофорами, зелений зазвичай асоціюється з безпекою, а червоний – з небезпекою. Розширення використовує цю впізнавану колірну схему, щоб зробити зміст своїх повідомлень зрозумілим.

Рисунок 3.3 – Демонстрація інтерфейсу програми в різних варіаціях

Після того, як прототип розширення було розроблено з використанням цієї колірної схеми, скріншоти прототипу були протестовані за допомогою «Coblis», онлайн-інструменту, який дозволяє користувачеві переглядати завантажене зображення так, як його побачила б людина з порушенням кольорового зору. Цей інструмент був використаний для того, щоб розширення браузера було легко читати людині з різними типами порушень зору.

Макет розширення був розроблений з урахуванням простоти, гарантуючи, що текст, зображення та кнопки були великими, чітко видимими та достатньо розташованими, щоб кожен елемент розширення можна було побачити та легко натиснути.

Щоб вибрати електронний лист, який буде оцінено розширенням, користувач відкриває його у необробленому вигляді та натискає кнопку «аналізувати» у вікні розширення. Цей метод дозволяє проаналізувати окрім вмісту електронного листа також й необхідні для аналізу заголовки та приховані посилання.

Спливаюче вікно розширення відображається в правій частині веб-сторінки; це звичайна практика для розширень веб-браузера, тому є

очікуваним для користувача місцерозташуванням. Це також запобігає перебоям у роботі браузера користувача; враховуючи те, що користувач, імовірно, має вирівняний за лівим краєм текст на сторінці, спливаюче вікно не охоплюватиме жодних важливих частин тексту на веб-сторінці.

Макет і дизайн спливаючого вікна розширення веб-браузера було розроблено з використанням HTML і CSS, тоді як JavaScript використовувався для внесення змін у вміст спливаючого вікна (Рисунок 3.4).

Розширення складається з набору файлів, упакованих для розповсюдження та інсталяції.

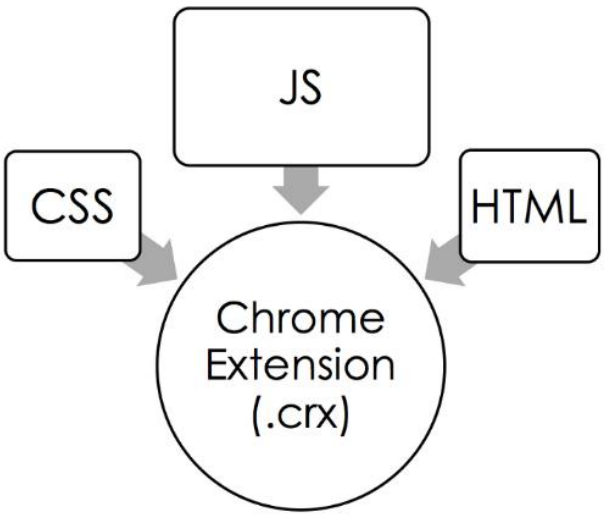


Рисунок 3.4 – Архітектура розширення Chrome

Розширення складається з файлу manifest.json, який має бути присутнім у кожному розширенні. Він містить основні метадані, такі як його ім'я, версію та необхідні дозволи. Він також надає вказівники інші файли в розширенні.

Маніфест містить наступні вказівники на кілька інших типів файлів (рисунок 3.5):

– Background page : Реалізує довготривалу логіку.
– Іконки для розширення

– popups: HTML-документ, які надають вміст для різних компонентів інтерфейсу користувача.

– Content scripts: JavaScript сценарії розширення, які будуть виконуватися на веб-сторінках.

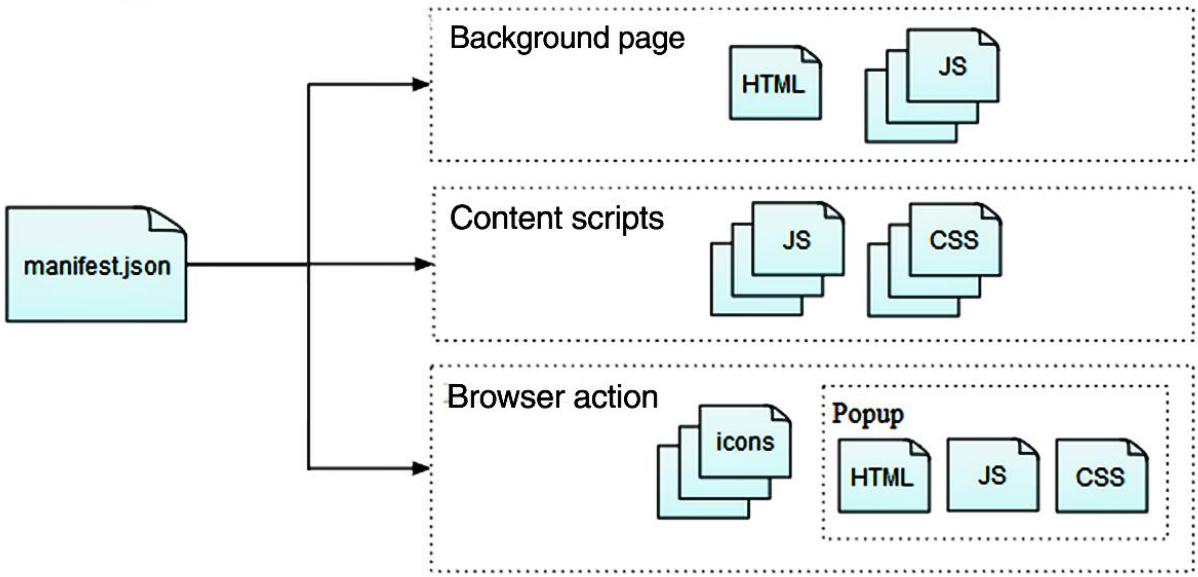


Рисунок 3.5 – Архітектура розширення Chrome – Athena logic

Для полегшення використання Athena Logic до розширення була додана невелика інструкція з використання.

Як вже було зазначено, для аналізу електронного листа, необхідно відкрити його у необробленому вигляді. Інструкція містить пояснення, як отримати оригінали листів (рисунок 6) на прикладах одних з найпоширеніших поштових сервісів України [49]:

– Gmail
– Outlook
– Yahoo mail
– Ukr.net

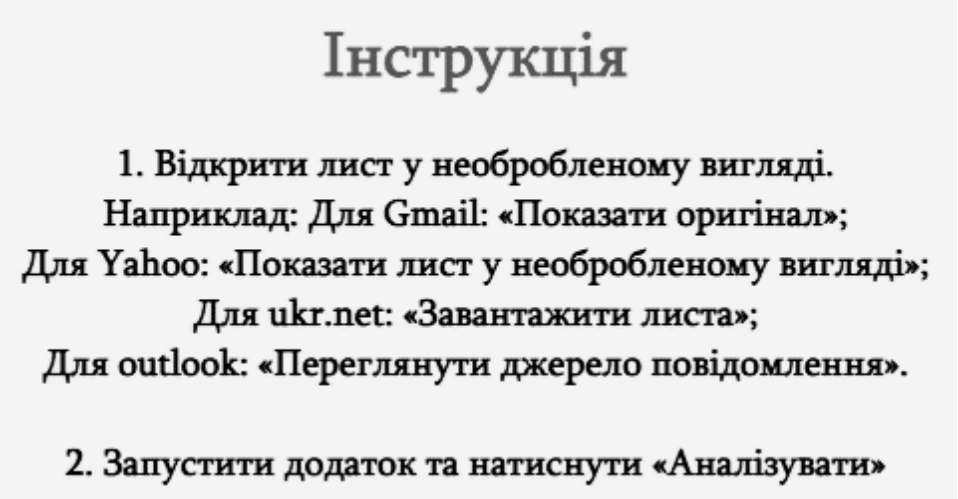


Рисунок 3.6 – Фрагмент сторінки інструкції

Також сторінка з інструкцією містить рекомендовані дії при отриманні результату аналізу «Сумнівно» або «Небезпечно».

Результат «Сумнівно» був доданий для запобіганню помилково хибних спрацювань. Але при отриманні даного результату, рекомендовано:

– не переходити за посиланнями та не відкривати жодних вкладень, якщо відправник невідомий;

– перевірити, чи немає "помилки" у веб-адресі;

– перевірити походження сторінки, перш ніж розміщувати на ній свою особисту інформацію - ім'я користувача та поштову адресу, фінансову інформацію тощо;

– якщо виникли сумніви щодо введення своїх облікових даних на підозрілому сайті, негайно змінити пароль.

3.4 Дослідження ефективності розширення

Для виявлення ефективності даного розширення на практиці, було проведене дослідження, у якому порівнювались п’ять інших розширень (Таблиця 3.1) з Athena logic.

Таблиця 3.1 – Огляд інших додатків для Chrome, обраних для дослідження

Розширення	Опис	Переваги	Недоліки
MailTrout	Використовується метод на основі машинного навчання	Зручність використання; Розширення подається як освітній інструмент, який навчає користувачів самостійно виявляти фішингові електронні листи	Хибні спрацювання; великий час оброблень
Ter7AntiPhishing	Семантичний аналіз з використанням словника	Зручність використання	Малий відсоток вірно визначених фішингових листів; Лише для Gmail
PhishBlock Prediction	Аналізує url-адреси в листах Евристичний метод	Швидкий аналіз	Наявність певного відсотка помилково-позитивних результатів; Не переглядає вміст електронної пошти; Лише для Gmail
Detect spam emails	Семантичний аналіз з використанням словника	Зручність використання	Хибні спрацювання
Email Phishing Tool	Семантичний аналіз з використанням словника Аналізує url-адреси в листах Евристичний метод	Хороші результати на невеликих наборах даних.	Малий відсоток вірно визначених фішингових листів; Малий відсоток визначення фішингових url;

Загалом для тестування розширення та доведення ефективності гібридного аналізу, було проаналізовано 50 електронних листів, 15 з який були безпечними, а 35 – фішинговими. Приклад одного з листів зображений на рисунку 3.7.

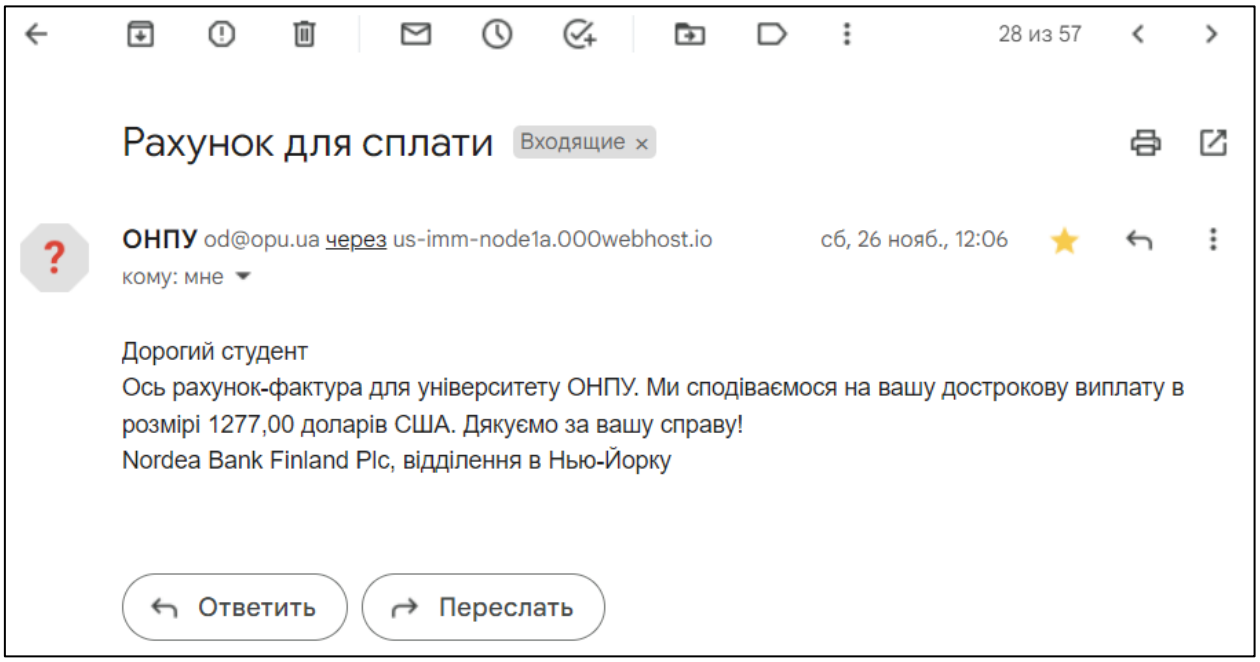


Рисунок 3.7 – Приклад листа, використаного під час дослідження

Таблиця 3.2 – Результати дослідження

Розширення	Точність, %
MailTrout	48
Ter7AntiPhishing	31
PhishBlock Prediction	54
Detect spam emails	30
Email Phishing Tool	42
Athena Logic	94

Результати дослідження, представлені у таблиці 3.2, показали ефективність Athena logic, та довели, що для успішного аналізу, необхідно

охоплювати обидві частини листа. Так, розширення Ter7AntiPhishing та Detect spam emails показали низький відсоток точних результатів через залежність від словникового методу: листи з коротким та, на перший погляд, легітимним змістом, але з небезпечними посиланнями, вони виявили як безпечні.

Розширення PhishBlock Prediction безглузде у випадках відсутності у листах посилань. MailTrout у свою чергу сприймає легітимні листи за фішингові.

Отже, була розроблена програмна реалізація методу, описаного у другому розділі, у вигляді розширення для браузера Chrome. Розширення браузера можуть діяти як доступні інструменти безпеки, оскільки для використання вони потребують небагато технічних знань і можуть бути легко включені в стандартну онлайн-діяльність людини.

Завдяки своїй простоті створене розширення може бути особливо корисними для тих, хто має менший досвід користування Інтернетом, і як інструменти безпеки можуть ефективно захистити найбільш вразливі групи.

Було проведене дослідження, метою якого було порівняти вже існуючі розширення для захисту пошти від фішингових атак, які використовують різні методи аналізу, з запропонованим методом цієї роботи. Результати продемонстрували ефективність розробленого розширення.

ВИСНОВКИ

У результаті виконання магістерської атестаційної роботи: була. доведена актуальність обраної теми, обґрунтована мета розробки вдосконаленого методу, та поставлені задачі для досягнення виконання мети.

У ході роботи були виконані наступні задачі:

1. Був проведений аналіз існуючих методів з забезпечення безпеки інформаційно-телекомунікаційних мереж від фішингових атак: визначені недоліки цих методів.
2. Були розглянуті основні критерії оцінювання ступеня небезпеки електронного листа, та на їх основі був розроблений алгоритм з використанням комплексного методу з забезпечення безпеки інформаційно-телекомунікаційних мереж;
3. Для практичного застосування алгоритму було розроблене програмне забезпечення, у вигляді розширення для веб-браузера Google Chrome
4. Було проведене дослідження, яке доказало ефективність алгоритму та додатку, порівнюючи його з іншими існуючими засобами виявлення фішингу у електронній пошті.

За результатами магістерської наукової роботи можна зробити висновок, що поставлена мета – розробка алгоритму та програмної реалізації захисту електронної пошти від фішингових атак, яка може бути імплантована у сучасний браузер, була успішно виконана.

ПЕРЕЛІК ПОСИЛАНЬ

1. Cisco Secure. Cisco Secure Email Phishing Defense. URL: https://www.cisco.com/c/dam/en/us/products/collateral/security/cloud-email-security/at-a-glance-c45-740894.pdf.

2. Microsoft 365. Microsoft Defender для Office 365. URL: https://learn.microsoft.com/ru-ru/microsoft-365/security/office-365-security/defender-for-office-365?view=o365-worldwide

3. Boyle P., Shepherd L. A. MailTrout: A Machine Learning Browser Extension for Detecting Phishing Emails. *34th British HCI Conference*, London, UK. 2021. P. 104–115.

4. Reading Between the Lines: Content-Agnostic Detection of Spear-Phishing Emails / H. Gascon. Research in Attacks, Intrusions, and Defenses. Cham, 2018. С. 69–91. URL: https://doi.org/10.1007/978-3-030-00470-5_4.

5. НКЦК: у 2021 році в Україні зафіксовано вже майже 14 мільйонів інцидентів у сфері кібербезпеки. Рада національної безпеки і оборони України. URL: https://www.rnbo.gov.ua/ua/Diialnist/4797.html

6. Avanan. Global-Phish-Report. URL: https://www.avanan.com/hubfs/2019-Global-Phish-Report.pdf

7. Jansson K., von Solms R. Phishing for phishing awareness. *Behaviour & Information Technology*. 2013. V. 32, No 6. P. 584–593. URL: https://doi.org/10.1080/0144929x.2011.632650

8. Phishing Attacks: A Recent Comprehensive Study and a New Anatomy / Z. Alkhalil. Frontiers in Computer Science. 2021. URL: https://doi.org/10.3389/fcomp.2021.563060

9. The Radicati Group. Email Statistics Report, 2016-2020. URL: https://www.radicati.com/wp/wp-content/uploads/2016/01/Email_Statistics_Report_2016-2020_Executive_Summary.pdf

10. APWG. *Phishing Activity Trends Report*, 2nd Quarter 2022. URL: https://docs.apwg.org/reports/apwg_trends_report_q2_2022.pdf.

11. BBC News. Global ransomware attack causes turmoil. URL: https://www.bbc.com/news/technology-40416611.

12. Solon O. 'Petya' ransomware attack: what is it and how can it be stopped?. The Guardian. URL: https://www.theguardian.com/technology/2017/jun/27/petya-ransomware-cyber-attack-who-what-why-how.

13. Kharpal A. ‘Petya’ ransomware: All you need to know about the cyberattack and how to tell if you’re at risk. *CNBC*. URL: https://www.cnbc.com/2017/06/28/petya-ransomware-cyberattack-explained-how-to-tell-if-youre-at-risk-or-been-attacked.html.

14. Greenberg A. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. *Wired*. URL: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/.

15. Капустинська К. Через атаку вірусу Petya Україна за півгодини втратила 10 млрд гривень – експерт. *Сьогодні*. URL: https://economics.segodnya.ua/ua/economics/enews/iz-za-ataki-virusa-petya-ukraina-za-polchasa-poteryala-10-mlrd-griven-ekspert-1069181.html.

16. Економічна правда. РНБО запровадила санкції проти причетних до розробки вірусу Petya. Українська правда. URL: https://www.epravda.com.ua/news/2021/06/18/675170/.

17. Maynor D., Olney M., Younan Y. The MeDoc Connection. URL: https://blog.talosintelligence.com/the-medoc-connection/.

18. Polityuk P., Prentice A. Exclusive: Ukraine hit by stealthier phishing attacks during BadRabbit strike. Reuters. URL: https://www.reuters.com/article/us-cyber-summit-ukraine-police-exclusive/exclusive-ukraine-hit-by-stealthier-phishing-attacks-during-badrabbit-strike-idUSKBN1D2263.

19. Служба безпеки України. СБУ закликає не скачувати програми із шахрайських розсилок. URL: https://ssu.gov.ua/novyny/sbu-zaklykaie-ne-skachuvaty-prohramy-iz-shakhraiskykh-rozsylok.

20. Nakashima E., Harris S. How the Russians hacked the DNC and passed its emails to WikiLeaks. The Washington Post. URL: https://ssu.gov.ua/novyny/sbu-zaklykaie-ne-skachuvaty-prohramy-iz-shakhraiskykh-rozsylok.

21. Computer Emergency Response Team of Ukraine. Кібератака на державні організації з використанням тематики іранських дронів-камікадзе Shahed-136 та шкідливої програми DolphinCape (CERT-UA#5683). URL: https://cert.gov.ua/article/3192088.

22. Держспецзв'язку. Виявлено масові фішингові листи на email-адреси українських сервісів i.ua та meta.ua. Telegram. URL: https://t.me/dsszzi_official/933.

23. Using automated individual white-list to protect web digital identities / W. Han та ін. Expert systems with applications. 2012. Т. 39, № 15. С. 11861–11869. URL: https://doi.org/10.1016/j.eswa.2012.02.020

24. Jain A. K., Gupta B. B. A novel approach to protect against phishing attacks at client side using auto-updated white-list. *EURASIP journal on information security*. 2016. Vol. 2016, No. 1. URL: https://doi.org/10.1186/s13635-016-0034-3

25. Kührer M., Holz T. An empirical analysis of malware blacklists. *PIK - Praxis der Informationsverarbeitung und Kommunikation.* 2012. Vol. 35, No. 1. URL: https://doi.org/10.1515/pik-2012-0003

26. Rao R. S., Vaishnavi T., Pais A. R. CatchPhish: detection of phishing websites by inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*. 2019. V. 11, No 2., P. 813–825. URL: https://doi.org/10.1007/s12652-019-01311-4

27. URL2Vec: URL modeling with character embeddings for fast and accurate phishing website detection / H. Yuan et.al. *IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/ SocialCom/ SustainCom)*, Melbourne, Australia. 2018. URL: https://doi.org/10.1109/bdcloud.2018.00050

28. Online deep learning: learning deep neural networks on the fly / D. Sahoo et al. *Twenty-Seventh international joint conference on artificial intelligence {IJCAI-18}*, Stockholm, Sweden, 13–19 July 2018. California, 2018. URL: https://doi.org/10.24963/ijcai.2018/369

29. Reading Between the Lines: Content-Agnostic Detection of Spear-Phishing Emails / H. Gascon et al. *Research in Attacks, Intrusions, and Defenses*. Cham, 2018. P. 69–91. URL: https://doi.org/10.1007/978-3-030-00470-5_4

30. Orunsolu A. A., Sodiya A. S., Akinwale A. T. A predictive model for phishing detection. *Journal of King Saud University. Computer and Information Sciences.* 2019. URL: https://doi.org/10.1016/j.jksuci.2019.12.005

31. Klensin J. Simple Mail Transfer Protocol. *RFC Editor*, 2008. URL: https://doi.org/10.17487/rfc5321.

32. RFC 4871 DomainKeys Identified Mail (DKIM) Signatures -- Update / ed. by D. Crocker. *RFC Editor*, 2009. URL: https://doi.org/10.17487/rfc5672.

33. Ukrainian tone dictionary. *GitHub*. URL: https://github.com/lang-uk/tone-dict-uk (date of access: 15.12.2022).

34. H. Manguri K., N. Ramadhan R., R. Mohammed Amin P. Twitter Sentiment Analysis on Worldwide COVID-19 Outbreaks. *Kurdistan Journal of Applied Research.* 2020. P. 54–65. URL: https://doi.org/10.24017/covid.8.

35. Renahan M. The Ideal Length of a Sales Email, Based on 40 Million Emails. *HubSpot Blog*. URL: https://blog.hubspot.com/sales/ideal-length-sales-email.

36. Kristensen E. What's the Ideal Email Length?. *The Ecommerce Revenue Engine | Drip*. URL: https://www.drip.com/blog/ideal-email-length.

37. Verizon. Data Breach Investigations Report. URL: https://www.knowbe4.com/hubfs/rp_DBIR_2017_Report_execsummary_en_xg.pdf.

38. Orunsolu A. A., Sodiya A. S., Akinwale A. T. A predictive model for phishing detection. *Journal of King Saud University - Computer and Information Sciences*. 2019. URL: https://doi.org/10.1016/j.jksuci.2019.12.005.

39. Warwick Ashford. Google launches beta version of Chrome web browser. *Computer Weekly*. URL: https://archive.ph/20210411080810/https://www.computerweekly.com/news/2240086779/Google-launches-beta-version-of-Chrome-web-browser.

40. W3Counter. Browser & Platform Market Share 2022. URL: https://web.archive.org/web/20221018205527/https://www.w3counter.com/globalstats.php.

41. StatCounter. Desktop Browser Market Share Ukraine. 2022. URL: https://gs.statcounter.com/browser-market-share/desktop/ukraine.

42. Welcome to Chrome Extensions - Chrome Developers. Chrome Developers. URL: https://developer.chrome.com/docs/extensions/mv3/.

43. Rakowski B. Google Chrome for the holidays: Mac, Linux and extensions in beta. *Google Chrome Blog*. URL: https://chrome.googleblog.com/2009/12/ google-chrome-for-holidays-mac-linux.html.

44. Google Chrome Help. Explore Google Chrome features: What are extensions? URL: https://web.archive.org/web/20100208233953/ https://www.google.com/support/chrome/bin/answer.py?answer=154007.

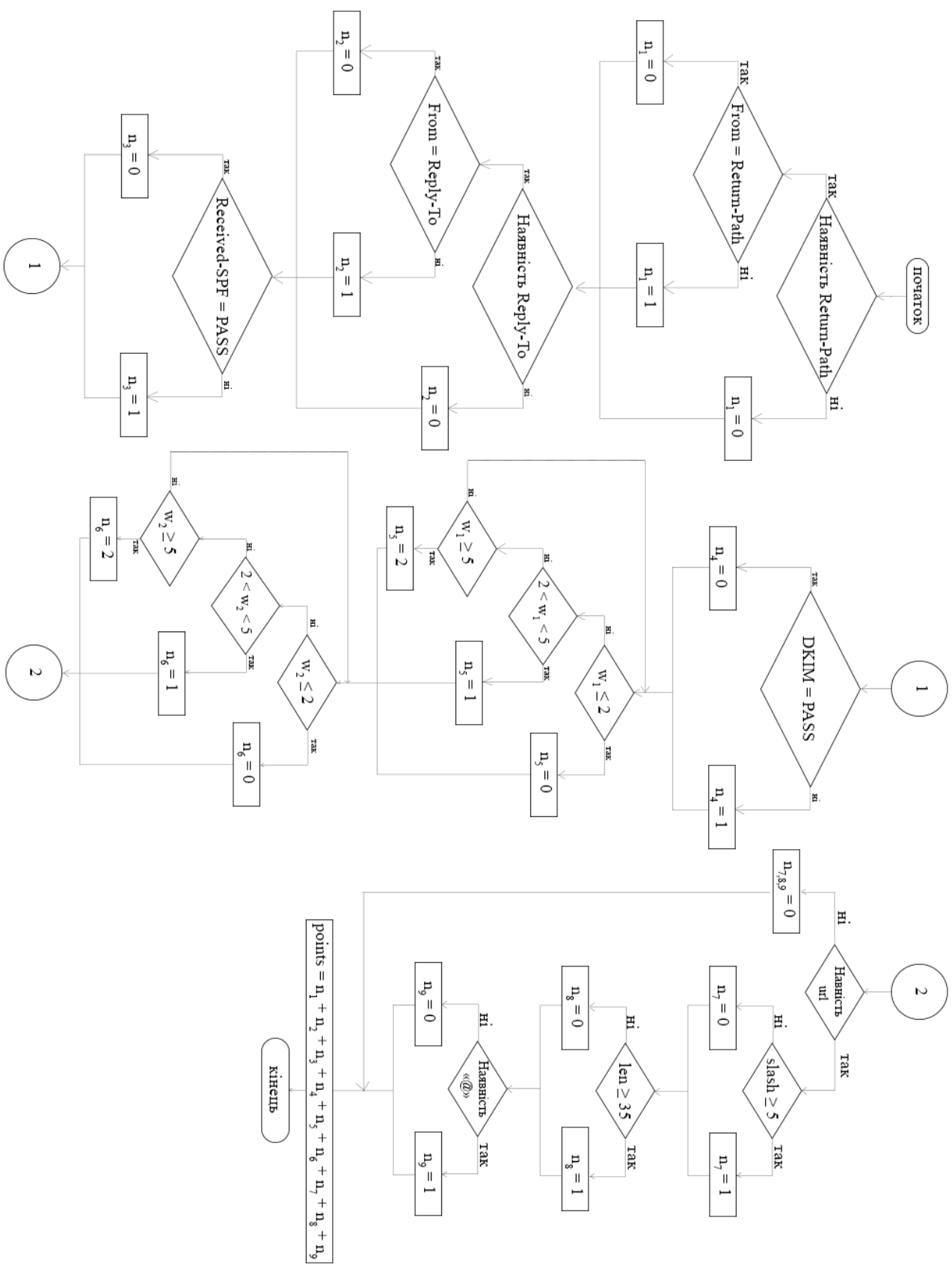
45. Boodman A. Extensions Status: On the Runway, *Getting Ready for Take-Off*. *Chromium Blog*. URL: https://blog.chromium.org/2009/09/extensions-status-on-runway-getting.html.

46. Dunlop M., Groat S., Shelly D. GoldPhish: Using Images for Content-Based Phishing Analysis. *2010 Fifth International Conference on Internet Monitoring and Protection, Barcelona, Spain*, 2010. URL: https://doi.org/10.1109/icimp.2010.24.

47. JetBrains. Meet WebStorm | WebStorm. WebStorm Help. URL: https://www.jetbrains.com/help/webstorm/meet-webstorm.html#first-steps.

48. ТОП 5 електронних поштових сервісів України. *Marketer*. URL: https://web.archive.org/web/20221212225549/https://marketer.ua/ua/top-5-email-services-in-ukraine/.

ДОДАТОК А. Алгоритм аналізу електронного листа



ДОДАТОК Б. Лістинг коду

```
chrome.runtime.onMessage.addListener(
    function(request, sender, sendResponse, Points) {
        let totalpoints = 0;
        function Headers() {
            let points = 0;
            try {
                let string = document.body.innerHTML;
                let  someText  =  string.replace(/(\r\n|\n|\r)/gm,
" ");

                let lis_from = ["From:"];
                let lis_received_spf = ["Received-SPF:"];
                let lis_domain = ["of"];
                let lis_return_path_a = ["Return-Path:"];
                let lis_return_path_b = ["Return-path:"];
                let lis_reply_to = ["Reply-To:"]

                function From() {
                    function From_one(string, lst_) {
                        let tmp_lst = someText.split(" ");
                        let res = [];
                        let N = lst_.length;

                        for (let i = 0; i < tmp_lst.length - N;
i++) {

                            if     (lis_from.join('     ')     ==
tmp_lst.slice(i, i + N).join(' ')) {
                                res.push(tmp_lst[i + 2])
                            }
                        }

                        return res;


                    }

                    let from_one = From_one(someText, lis_from);
                    from_a = from_one.toString()

                    function From_two(string, lst_) {
                        let tmp_lst = someText.split(" ");
                        let res = [];
                        let N = lst_.length;

                        for (let i = 0; i < tmp_lst.length - N;
i++) {
```

```
                                if     (lis_from.join('     ')     ==
tmp_lst.slice(i, i + N).join(' ')) {
                                    res.push(tmp_lst[i + 3])
                                }
                            }

                            return res;


                        }

                        let from_two = From_two(someText, lis_from);
                        from_b = from_two.toString()

                        a = from_a.includes(";")
                        if (a != false) {
                            b = from_a
                        } else {
                            b = from_b
                        }

                        function delete_sym_from() {
                            var ret_c = b.toString();
                            let ret_a = ret_c.includes(";")

                            if (ret_a != false) {
                                a = ret_c.slice(4, -4)
                            }
                            return a;

                        }

                        let from = delete_sym_from();

                        return from
                    }

                    function Received_spf(someText, lst_) {
                        let tmp_lst = someText.split(" ");
                        let res = [];
                        let N = lst_.length;

                        for (let i = 0; i < tmp_lst.length - N; i++)
{
                            if     (lis_received_spf.join('')     ==
tmp_lst.slice(i, i + N).join('')) {
                                res.push(tmp_lst[i + N]);
                            }
                        }
                        return res;
                    }
```

```
                function Domain(someText, lst_) {
                    let tmp_lst = someText.split(" ");
                    let res = [];
                    let N = lst_.length;

                    for (let i = 0; i < tmp_lst.length - N; i++)
{


                        if         (lis_domain.join('')         ==
tmp_lst.slice(i, i + N).join('')) {
                            res.push(tmp_lst[i + N]);
                        }


                    }
                    return res;
                }

                function Return_Path() {
                    function Return_Path_a(someText, lst_) {
                        let tmp_lst = someText.split(" ");
                        let res = [];
                        let N = lst_.length;

                        function f1() {
                            for (let i = 0; i < tmp_lst.length -
N; i++) {
                                if    (lis_return_path_a.join('')
== tmp_lst.slice(i, i + N).join('')) {
                                    res.push(tmp_lst[i + N]);
                                }

                            }
                            return res;
                        }

                        let return_path = f1();

                        var ret_c = return_path.toString();
                        let ret_a = ret_c.includes(";")
                        if (ret_a != false) {
                            return_path = ret_c.slice(4, -4)
                        }

                        return return_path;

                    }

                    function Return_Path_b(someText, lst_) {
```

```
                        let tmp_lst = someText.split(" ");
                        let res = [];
                        let N = lst_.length;

                        function f1() {
                            for (let i = 0; i < tmp_lst.length -
N; i++) {
                                if    (lis_return_path_b.join('')
== tmp_lst.slice(i, i + N).join('')) {
                                    res.push(tmp_lst[i + N]);
                                }

                            }
                            return res;
                        }

                        let return_path = f1();

                        var ret_c = return_path.toString();
                        let ret_a = ret_c.includes(";")
                        if (ret_a != false) {
                            return_path = ret_c.slice(4, -4)
                        }

                        return return_path;

                    }

                    let  return_path_a  =  Return_Path_a(someText,
lis_return_path_a);
                    let  return_path_b  =  Return_Path_b(someText,
lis_return_path_b);
                    let    return_path    =    return_path_a    +
return_path_b
                    return return_path
                }

                function Reply_To() {
                    function Reply_To_a(someText, lst_) {
                        function Reply_to_one(string, lst_) {
                            let tmp_lst = someText.split(" ");
                            let res = [];
                            let N = lst_.length;

                            for (let i = 0; i < tmp_lst.length -
N; i++) {

                                if   (lis_reply_to.join('   ')   ==
tmp_lst.slice(i, i + N).join(' ')) {
                                    res.push(tmp_lst[i + 2])
                                }
```

```
                }

                return res;


            }

            let             reply_to_one             =
Reply_to_one(someText, lis_from);
            reply_to_a = reply_to_one.toString()

            function Reply_to_two(string, lst_) {
                let tmp_lst = someText.split(" ");
                let res = [];
                let N = lst_.length;

                for (let i = 0; i < tmp_lst.length -
N; i++) {

                    if  (lis_reply_to.join('  ')  ==
tmp_lst.slice(i, i + N).join(' ')) {
                        res.push(tmp_lst[i + 3])
                    }
                }

                return res;


            }

            let             reply_to_two             =
Reply_to_two(someText, lis_from);
            reply_to_b = reply_to_two.toString()

            a = reply_to_a.includes(";")
            if (a != false) {
                b = reply_to_a
            } else {
                b = reply_to_b
            }

            function delete_sym_from() {
                var ret_c = b.toString();
                let ret_a = ret_c.includes(";")

                if (ret_a != false) {
                    a = ret_c.slice(4, -4)
                }
                return a;

            }
```

```
                        let reply_to = delete_sym_from();

                        return reply_to

                    }

                    let    reply_to    =    Reply_To_a(someText,
lis_reply_to);

                    function r() {
                        if (reply_to == "") {
                            return points = 0;
                        } else {
                            return reply_to;
                        }
                    }

                    a = r()
                    return a
                }

                let from = From()
                let   received_spf    =    Received_spf(someText,
lis_received_spf);
                let domain = Domain(someText, lis_domain);
                let return_path = Return_Path();
                let reply_to = Reply_To(someText, lis_reply_to);

//перевірка
                function From_Return_Path(from, return_path) {
                    if(domain.length==0) {
                        points = 1
                    }      else      if      (from.length      ==
return_path.length) {
                        points = 0
                    }    else    if    (return_path.length    ==
domain[0].length) {
                        points = 0
                    } else {
                        let return_a = return_path.split("@")[1]
                        let return_b = domain[0].split("@")[1]
                        if  (return_b.length  ==  return_a.length)
{
                            points = 0
                        } else {
                            points = 1
                        }
                    }
                    return points
```

```
                }

                function Control_spf_pass() {
                    a = document.body.innerHTML.split("Received-
SPF:")[1]
                    if (a.includes("pass")) {
                        points = 0
                    } else {
                        points = 1
                    }
                    return points
                }

                function From_Reply_To(from, reply_to) {
                    if (from.length == reply_to.length) {
                        points = 0
                    } else if (reply_to == "") {
                        points = 0
                    } else {
                        points = 1
                    }
                    return points

                }

                function DKIM() {
                    a                                                =
document.body.innerHTML.split("Authentication-Result")[1]
                    if (a.includes("dkim=pass")) {
                        points = 0
                    } else {
                        points = 1
                    }
                    return points
                }

                let               control_spf_pass               =
Control_spf_pass(someText, lis_domain);
                let  from_return_path  =  From_Return_Path(from,
return_path);
                let    from_reply_to    =    From_Reply_To(from,
reply_to)
                let dkim = DKIM()

                points  =  from_reply_to  +  from_return_path  +
control_spf_pass + dkim
                return Number(points)
            } catch {
                return points = 0
            }
        }
```

```
        let headers = Headers()
        function Dictionary() {
            try {
                function Base64OrNot() {
                    abs = document.body.innerHTML
                    if  (abs.includes("Content-Transfer-Encoding:
base64")) {
                        mas = []
                        let   a   =   abs.split("Content-Transfer-
Encoding: base64")[1];

                        function split() {
                            if (a.includes("\n\n")) {
                                b = a.split("\n\n")[1]
                                if (b.includes("<")) {
                                    return b.split("<")[0]
                                } else if (a.includes("--")) {
                                    return b.split("--")[0]
                                }
                            }
                        }

                        function u_atob(ascii) {
                            return   Uint8Array.from(atob(ascii),
c => c.charCodeAt(0));
                        }

                        var base64String2 = split()
                        utf                 =                 new
TextDecoder().decode(u_atob(base64String2))

                        return utf

                    }  else  if  (abs.includes("Content-Transfer-
Encoding: quoted-printable")) {
                        let   a   =   abs.split("Content-Transfer-
Encoding:  quoted-printable")[1]  +  abs.split("Content-Transfer-
Encoding: quoted-printable")[2]
                        return a
                    } else if(abs.includes("<!DOCTYPE html>")){
                        let b = abs.split("<!DOCTYPE html>")
                        return b
                    }

                }

                text = Base64OrNot()

                function     Tone_dictionary(request,     sender,
sendResponse, search) {
                    const searches_en = tone_en;
```

```
                    const searches_ua = tone_ua;          const
searches_ru = tone_ru;
                    let totalpoints = 0;
                    for (search of searches_ua) {
                        let re = new RegExp(search, 'gi')
                        let matches = text.match(re);
                        if (matches != null) {
                            totalpoints = totalpoints + 1;
                        }
                        for (search of searches_en) {
                            let re = new RegExp(search, 'gi')
                            let matches = text.match(re);
                            if (matches != null) {
                                totalpoints = totalpoints + 1;
                            }
                        }
                        for (search of searches_ru) {
                            let re = new RegExp(search, 'gi')
                            let matches = text.match(re);
                            if (matches != null) {
                                totalpoints = totalpoints + 1;
                            }
                        }

                        return totalpoints
                    }
                }

                function     Spam_dictionary(request,     sender,
sendResponse, search) {
```

```
const searches_en = ["#1", "100% more", "100% free", "100% satisfied", "Additional income", "Be your own boss", "Best price", "Big bucks", "Billion", "Cash bonus", "Cents on the dollar", "Consolidate debt", "Double your cash", "Double your income", "Earn extra cash", "Earn money", "Eliminate bad credit", "Extra cash", "Extra income", "Expect to earn", "Fast cash", "Financial freedom", "Free access", "Free consultation", "Free gift", "Free hosting", "Free info", "Free investment", "Free membership", "Free money", "Free preview", "Free quote", "Free trial", "Full refund", "Get out of debt", "Get paid", "Giveaway", "Guaranteed", "Increase sales", "Increase traffic", "Incredible deal", "Lower rates", "Lowest price", "Make money", "Million dollars", "Miracle", "Money back", "Once in a lifetime", "One time", "Pennies a day", "Potential earnings", "Prize", "Promise", "Pure profit", "Risk-free", "Satisfaction guaranteed", "Save big money", "Save up to", "Special promotion", "Act now", "Apply now", "Become a member", "Call now", "Click below", "Click here", "Get it now", "Do it today", "Don't delete", "Exclusive deal", "Get started now", "Important information regarding", "Information you requested", "Instant", "Limited time", "New customers only", "Order now", "Please read", "See for yourself", "Sign up free", "Take action", "This won't last", "Urgent", "What are you waiting for?", "While supplies last", "Will not believe your eyes", "Winner", "Winning", "You are a winner", "You have been selected", "Bulk email", "Buy direct", "Cancel at any time", "Check or money order", "Congratulations", "Confidentiality", "Cures", "Dear friend", "Direct email", "Direct marketing", "Hidden charges", "Human growth hormone", "Internet marketing", "Lose weight", "Mass email", "Meet singles", "Multi-level marketing", "No catch", "No cost", "No credit check", "No fees", "No gimmick", "No hidden costs", "No hidden fees", "No interest", "No investment", "No obligation", "No purchase necessary", "No questions asked", "No strings attached", "Not junk", "Notspam", "Obligation", "Passwords", "Requires initial investment", "Social security number", "This isn't a scam", "This isn't junk", "This isn't spam", "Undisclosed", "Unsecured credit", "Unsecured debt", "Unsolicited", "Valium", "Viagra", "Vicodin", "We hate spam", "Weight loss", "Xanax", "Accept credit cards", "Ad", "All new", "As seen on", "Bargain", "Beneficiary", "Billing", "Bonus", "Cards accepted", "Cash", "Certified", "Cheap", "Claims", "Clearance", "Compare rates", "Credit card offers", "Deal", "Debt", "Discount", "Fantastic", "In accordance with laws", "Income", "Investment", "Join millions", "Lifetime", "Loans", "Luxury", "Marketing solution", "Message contains", "Mortgage rates", "Name brand", "Offer", "Online marketing", "Opt in", "Pre-approved", "Quote", "Rates", "Refinance", "Removal", "Reserves the right", "Score", "Search engine", "Sent in compliance", "Subject to…", "Terms and conditions", "Trial", "Unlimited", "Warranty", "Web traffic", "Work from home"]
```

```
const searches_ua = ["# один", "100% безкоштовно", "50% знижка", "0% ризик", "Акції", "Банкрутство", "Без вкладень", "Без зобов'язань", "Без зобов'язань", "Без досвіду", "Без ризику", "Безлімітний", "Безвідсотковий кредит", "Безризиковий", "Безкоштовна консультація", "Безкоштовна установка", "Безкоштовно", "Безкоштовний доступ", "Безкоштовний сайт", "Безкоштовний тестовий доступ", "Бізнес на дому", "Бонус", "Бренд", "В день", "в тиждень", "в місяць", "Ваш статус", "Віагра", "Вклад", "Повернення грошей", "Ви були обрані", "Ви переможець", "Вигідна угода", "Виграш", "Виплата", "Гарантії", "Гарантований", "Гарантія повернення грошей", "Дебет", "кредит", "Дій зараз", "Гроші", "Дешева іпотека", "Дешево", "Діагностика", "Борг", "Долар", "будинок", "Домашній бізнес", "Домен", "Додатковий дохід", "Дорогий", "друг", "Доступ", "Доступний", "Дохід", "Друг", "Замовлення", "замовити", "Запусти свій бізнес", "Заробляй $", "Заробляй удома", "Зароби", "Заробіток", "Зареєструйтесь", "Зареєструйтесь зараз", "Навіщо платити більше", "Тут", "Знаменитість", "Позбудьтеся", "Лікуй", "Інструкції", "Інформація на ваш запит", "Іпотека", "Казино", "Мапа", "Колектори", "Комерційна пропозиція", "Комерція", "Конкурс", "Конфіденційно", "Копія", "Кредит", "Кредитне бюро", "Кредитні картки", "Кредитори", "Купити безпосередньо", "Придбати", "Ліки", "Кращу пропозицію", "Маркетинг", "Мільйон доларів", "Мільйон", "Мільйонер", "Мінімальний платіж", "Безліч", "Натисніть тут", "Натисни сюди", "Назва бренду", "Готівка", "Почни зараз", "Не зволікай", "Не проходьте повз", "Не спам", "Не видаляйте", "Необмежені можливості", "Необмежений доступ", "Новинка", "новий", "Номер один", "Зразок", "Відкрийте", "Перейдіть за посиланням", "Персональна знижка", "Харчові добавки", "Перемога", "переможець", "Передплата", "Підпишіться", "Зателефонуйте", "Вітаємо", "Покупка", "Одержувач", "Отримай мільйон", "Отримай зараз", "Порно", "Завітайте", "наш сайт", "Останній шанс", "Приголомшливий", "Схуднення", "Прайс", "Пропозиція обмежена", "Перевага", "вітання", "Приз (подарунок)", "Продажі", "Процентна ставка", "Прочитайте", "Працюй з дому", "Роздрукуйте", "Розсилка", "Результат", "Реклама", "Рефінансування", "Рішення", "Розіграш", "Ринок", "Сам собі начальник", "Сама", "низька ціна", "Найнижчі ціни", "Заощадження", "Скиньте вагу", "Свобода", "Зроби це зараз", "Угода", "Секрет успіху", "Секс", "Сертифікований", "Знижка", "Приховані комісії", "Прихований", "Збережіть", "Спеціальна знижка", "Спеціально для Вас", "Спеціальна пропозиція", "Спеціальний засіб", "Порівняйте", "Середній", "Термін", "Ставки на спорт", "Ставки", "Статус замовлення", "Сто відсотків", "Сто відсотків безкоштовно", "Стоп", "Телефон", "Тест", "Тільки зараз", "Транзакції", "Тисяча", "Збільшити трафік", "Задоволення", "Насолода 100%", "Умови", "Успіх", "Фантастичний", "Форма", "Фріланс", "Хропіння", "Чудо", "Шанс", "Ексклюзив"]
```

```
                    const searches_ru = ["# один", "50% скидка",
"0% риск", "Акции", "Банкротство", "Без вложений", "Без
обязательств", "Без обязательств", "Без опыта", "Без риска",
"Безлимитный", "Беспроцентный кредит", "Безрисковый",
"Бесплатная консультация", "Бесплатная установка", "Бесплатно",
"Бесплатный доступ", "Бесплатный сайт", "Бесплатный тестовый
доступ", "Бизнес на дому", "Бонус", "Бренд", "В день", "в
неделю", "в месяц", "Ваш статус", "Виагра", "Вклад", "Возврат
денег", "Вы были выбраны", "Вы победитель", "Выгодная сделка",
"Выигрыш", "Выплата", "Гарантии", "Гарантированный", "Гарантия
возврата денег", "Дебет", "кредит", "Действуй сейчас", "Деньги",
"Дешевая ипотека", "Дешево", "Диагностика", "Долг", "Доллар",
"Дом", "Домашний бизнес", "Домен", "Дополнительный доход",
"Дорогой", "друг", "Доступ", "Доступный", "Доход", "Друг",
"Заказ", "заказать", "Запусти свой бизнес", "Зарабатывай $",
"Зарабатывай дома", "Заработай", "Заработок",
"Зарегистрируйтесь", "Зарегистрируйтесь сейчас", "Зачем платить
больше", "Здесь", "Знаменитость", "Избавьтесь", "Излечени",
"Инструкции", "Информация по вашему запросу", "Ипотека",
"Казино", "Карта", "Коллекторы", "Коммерционное предложение",
"Коммерция", "Конкурс", "Конфиденциально", "Копия", "Кредит",
"Кредитное бюро", "Кредитные карты", "Кредиторы", "Купить
напрямую", "Купить", "Лекарства", "Лучшее предложение",
"Маркетинг", "Миллион долларов", "Миллион", "Миллионер",
"Минимальный платеж", "Множество", "Нажми здесь", "Нажми сюда",
"Название бренда", "Наличные", "Начни сейчас", "Не медли", "Не
проходите мимо", "Не спам", "Не удаляйте", "Неограниченные
возможности", "Неограниченный доступ", "Новинка", "Новый",
"Номер один", "Образец", "Откройте", "Перейдите по ссылке",
"Персональная скидка", "Пищевые добавки", "Победа",
"победитель", "Подписка", "Подпишитесь", "Позвоните",
"Поздравляем", "Покупка", "Получатель", "Получи миллион",
"Получи сейчас", "Порно", "Посетите", "наш сайт", "Последний
шанс", "Потрясающий", "Похудение", "Прайс", "Предложение
ограниченно", "Преимущество", "Привет", "Приз (подарок)",
"Продажи", "Процентная ставка", "Прочитайте", "Работай из дома",
"Распечатайте", "Рассылка", "Результат", "Реклама",
"Рефинансирование", "Решение", "Розыгрыш", "Рынок", "Сам себе
начальник", "Самая", "низкая цена", "Самые низкие цены",
"Сбережения", "Сбросьте вес", "Свобода", "Сделай это сейчас",
"Сделка", "Секрет успеха", "Секс", "Сертифицированный",
"Скидка", "Скрытые комиссии", "Скрытый", "Сохраните",
"Специальная скидка", "Специально для вас", "Специальное
предложение", "Специальное средство", "Сравните", "Средний",
"Срок", "Ставки на спорт", "Ставки", "Статус заказа", "Сто
процентов", "Сто процентов бесплатно", "Стоп", "Телефон",
"Тест", "Только сейчас", "Транзакции", "Тысяча", "Увеличь
трафик", "Удовлетворение", "Удовольствие 100%", "Условия",
"Успех", "Фантастический", "Форма", "Фриланс", "Храп", "Чудо",
"Шанс", "Эксклюзив"]
                    let totalpoints = 0;
```

```
                    for (search of searches_ua) {
                        let re = new RegExp(search, 'gi')
                        let matches = text.match(re);
                        if (matches != null) {
                            totalpoints = totalpoints + 1;
                        }
                        for (search of searches_en) {
                            let re = new RegExp(search, 'gi')
                            let matches = text.match(re);
                            if (matches != null) {
                                totalpoints = totalpoints + 1;
                            }
                        }
                        for (search of searches_ru) {
                            let re = new RegExp(search, 'gi')
                            let matches = text.match(re);
                            if (matches != null) {
                                totalpoints = totalpoints + 1;
                            }
                        }
                        return totalpoints
                    }
                }

                spam_dictionary = Spam_dictionary()
                tone_dictionary = Tone_dictionary()

                function Points() {
                    let points = 0;

                    function Tone() {
                        if (tone_dictionary >= 5) {
                            points = 2;
                        }                   else                    if
((tone_dictionary>2)&&(tone_dictionary<5)){
                            points = 1
                        }
                        else {
                            points = 0;
                        }
                        return points
                    }

                    function Spam() {
                        if (spam_dictionary >= 10) {
                            points = 2;
                        }else                                       if
((spam_dictionary>2)&&(spam_dictionary<10)) {
                            points = 1
                        }
                        else {
```

```
                            points = 0;
                        }
                        return points
                    }

                    let tone = Number(Tone());
                    let spam = Number(Spam())
                    points = tone + spam
                    return points;
                }

                points = Points()
                return points
            } catch {
                return points = 0
            }
        }
        let dictionary = Dictionary()
        function Url() {
            try{
                function Base64OrNot() {
                    abs = document.body.innerHTML
                    if  (abs.includes("Content-Transfer-Encoding:
base64")) {
                        let   a   =   abs.split("Content-Transfer-
Encoding: base64")[1]
                        if (a.includes("--")) {
                            var b = a.split("--")[0]
                        }

                        function u_atob(ascii) {
                            return   Uint8Array.from(atob(ascii),
c => c.charCodeAt(0));
                        }

                        var base64String2 = b
                        utf                 =                 new
TextDecoder().decode(u_atob(base64String2))
                        s = utf.split(" ")
                        mas = []

                        function str() {
                            for (var i = 1; i < s.length; i++) {
                                if (s[i].includes("/")) {
                                    mas.push(s[i])
                                }
                            }
                            return mas
                        }

                        url = str()
```

```
                        return url

                    }  else  if  (abs.includes("Content-Transfer-
Encoding:")) {
                        let   a   =   abs.split("Content-Transfer-
Encoding:")[1]
                        let b = a.split("</html>")[0]
                        mas = []
                        if (b.includes("href=3D")) {
                            c = b.split("href=3D")
                            for (var i = 1; i < c.length; i++) {
                                if (c[i].includes('"')) {
                                    mas.push(c[i].split('"
')[0])
                                }
                            }
                        }
                        return mas
                    } else if (abs.includes("!DOCTYPE")) {
                        let a = abs.split("!DOCTYPE")[1]
                        let b = a.split("</html>")[0]
                        mas = []
                        if (b.includes("href=3D")) {
                            c = b.split("href=3D")
                            for (var i = 1; i < c.length; i++) {
                                if (c[i].includes('"')) {
                                    mas.push(c[i].split('"
')[0])
                                }
                            }
                        }
                        return mas
                    }

                }
                utf = Base64OrNot()
                function Url_points() {
                    function Sym_one() {
                        points = 0
                        for (var i = 0; i < utf.length; i++) {
                            if (utf[i].includes("@")) {
                                points = 1;
                            }
                            return points
                        }
                    }
                    function Sym_two() {
                        points = 0
                        for (var i = 0; i < utf.length; i++) {
                            if (utf[i].length >= 35) {
                                points = 1
```

```
                                }
                                return points
                            }

                        }
                        sym_one = Sym_one()
                        sym_two = Sym_two()
                        function Slash() {

                            function length() {
                                mas = []
                                for (var i = 0; i < utf.length; i++)
{
                                    if   (utf[i].split("/").length   -
1) {

mas.push(utf[i].split("/").length - 1)
                                    }
                                }
                                return mas
                            }

                            len = length()

                            function One() {
                                points = 0
                                for (let i = 0; i < len.length; i++)
{
                                    if (len[i] >= 5) {
                                        points = 1
                                    }
                                }
                                return points
                            }

                            one = One()
                            return one
                        }

                        slash = Slash()
                        points = sym_one +sym_two + slash
                        return Number(points)
                    }

                    url = Url_points()
                    if (url >= 3) {
                        url = 3
                    } else if ((url>=2)&&(url<3)){
                        url = 2
                    } else if(url=1){
                        url = 1;
```

```
                } else{
                    url = 0
                }
                return url
            } catch {
                url = 0
                return url
            }
            }
            let url_points = Url()
            totalpoints = url_points + dictionary + headers

        sendResponse({count: totalpoints})
    }

);
```