Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
МЕТОД ТА ЗАСІБ ЗБИРАННЯ ІНФОРМАЦІЇ ПРО МІШЕНЬ ПІД ЧАС ІНФОРМАЦІЙНОЇ ВІЙНИ

Виконав: студент 2-го курсу, групи 1БС-21м
спеціальності 125 – Кібербезпека
(шифр і назва напряму підготовки, спеціальності)
_____ Хилько С. В.
(прізвище та ініціали)
Керівник к. т. н., доц., доц. каф. ЗІ
_____ Войтович О. П.
(прізвище та ініціали)
Опонент к.т.н., доц., доц. каф. ПЗ:
_____Майданюк В. П.
(прізвище та ініціали)
«19» 12 2022 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«19» 12 2022 р.

Вінниця ВНТУ – 2022 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Освітній ступінь магістр
Спеціальність 125 Кібербезпека
ОПП Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д. т. н., проф.
_____ В. А. Лужецький
21.09 _____ 2022 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Хильку Степану Вікторовичу

1. Тема роботи: «Метод та засіб збирання інформації про мішень під час інформаційної війни», керівник роботи: Войтович Олеся Петрівна, к. т. н., доц., доц. каф. ЗІ, затверджені наказом ректора ВНТУ 14 вересня 2022 року №203.
2. Строк подання студентом роботи 19 грудня 2022 р.
3. Вихідні дані до роботи:
   - можливість збирання інформації про конкретну особу;
   - Frontend-частина, яка представляє користувачам інформацію у зручному вигляді;
   - Backend-частина, яка відповідає за збирання, обробку, передачу інформації.
4. Зміст розрахунково-пояснювальної: Вступ. Аналіз інформаційних джерел. Розробка методу збирання інформації про мішень. Розробка алгоритму збирання інформації про мішень. Реалізація і тестування системи для збирання інформації про мішень під час інформаційної війни. Економічна частина. Висновки. Перелік інформаційних джерел. Додатки.
5. Перелік ілюстративного матеріалу.

Модель проведення атаки (плакат, А4). Системні модулі (плакат, А4). Алгоритм процесу збору інформації (плакат, А4). Загальна схема роботи системи (плакат, А4). Види інформації для проведення атак (плакат, А4). Результати тестування програмного засобу (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Войтович О. П., к. т. н., доц. каф. ЗІ		
2	Войтович О. П., к. т. н., доц. каф. ЗІ		
3	Войтович О. П., к. т. н., доц. каф. ЗІ		
4	Лесько О. Й., к. е. н., проф. каф. ЕПВМ		

7. Дата видачі завдання 1 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Прим
1	Аналіз завдання. Вступ	01.09.2022 – 04.09.2022	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2022 – 15.09.2022	
3	Науково-технічне обґрунтування	16.09.2022 – 22.09.2022	
4	Розробка технічного завдання	23.09.2022 – 04.10.2022	
5	Розробка методу, критеріїв оцінки інформаційних джерел	05.10.2022 – 24.10.2022	
6	Розробка алгоритму, програмна реалізація та тестування засобу для збирання інформації про мішень під час інформаційної війни	25.10.2022 – 17.11.2022	
7	Аналіз виконання ТЗ, висновки	18.11.2022 – 24.11.2022	
8	Оформлення пояснювальної записки	25.11.2022 – 30.11.2022	
9	Попередній захист та доопрацювання МКР	07.12.2022 – 15.12.2022	
10	Перевірка магістерської роботи на наявність плагіату	15.12.2022 – 16.12.2022	
11	Представлення МКР до захисту, рецензування	17.12.2022 – 20.12.2022	
12	Захист МКР	21.12.2022 – 23.12.2022	

Студент _____ С. В. Хи

Керівник роботи _____ О. П. Войто

АНОТАЦІЯ

УДК 004.056

Хилько С. В. Метод та засіб збирання інформації про мішень під час інформаційної війни. Магістерська кваліфікаційна робота зі спеціальності 125Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2022. 93 с.

На укр. мові. Бібліогр.: 26 назв; рис.: 44; табл. 12.

Магістерська кваліфікаційна робота присвячена розробці методу та засобу збирання інформації про мішень під час інформаційної війни. Здійснено аналіз сучасних методів та систем розвідки з використанням відкритих даних. Розроблено метод, що дозволяє покращити збирання інформації про мішень під час інформаційної війни. Розроблено програмний засіб, що дозволяє реалізувати запропонований метод. Виконано експериментальне дослідження запропонованого методу та засобу. Виконано обгрунтування економічної доцільності розробленого методу

Ілюстративна частина складається з 6 плакатів з демонстрацією результатів проведених досліджень та розробок.

Ключові слова: кібербезпека, інформаційна війна, OSINT, відкриті джерела

ABSTRACT

Khylko S. V. Method and means of gathering information about the target during the information warMaster's thesis in the specialty 125 – Cybersecurity. Vinnytsia: VNTU, 2022. 93 p.

In Ukrainian language. Bibliographer: 26 titles; fig .: 44; table. 12.

The master's thesis is devoted to the development of a method and means of gathering information about a target during an information war. An analysis of modern intelligence methods and systems using open data was carried out. A method has been developed to improve target information gathering during information warfare. A software tool has been developed that allows implementing the proposed method. An experimental study of the proposed method and tool was performed. The economic feasibility of the developed method has been substantiated.

The illustrative part consists of 6 posters with a demonstration of the results of research and development.

Keywords: cyber security, information warfare, OSINT, open sources

ЗМІСТ

ВСТУП

Сучасні атаки є надзвичайно складними операціями, для ефективного проведення яких аналізуються великі масиви даних, збирається багато персональної інформації про мішень, аналізуються соціальні мережі та перевіряється наявність інформації про мішень у злитих базах даних. Так, маючи такий обсяг даних про мішень, значно підвищуються шанси на успішну реалізацію атаки, адже вірогідність того, що мішень зреагує на тригери – значно вища.

Об’єктом дослідження є процес збирання інформації про мішень під час інформаційної війни.

Предметом є системи для збирання інформації про мішень під час інформаційної війни.

Метою магістерської кваліфікаційної роботи є покращення пошуку інформації про мішень, що дозволить реалізовувати інформаційні операції під час інформаційної війни.

Для досягнення мети необхідно виконати такі завдання:

- проаналізувати відкриті джерела інформації;
- проаналізувати системи для здійснення розвідки;
- розробити архітектуру системи;
- розробити алгоритм роботи системи;
- розробити програмну систему;
- виконати перевірку коректності роботи системи.

Наукова новизна магістерської роботи полягає в тому, що удосконалено пошук інформації про мішень під час інформаційної, яка на відміну від інших враховує етапи проведення інформаційної атаки та допомагає обрати джерела для збору інформації, що дозволяє підвищити ефективність роботи спеціалістів з кібербезпеки.

Практична цінність: алгоритм та програмний засіб для збору інформацій про мішень під час інформаційної війни.

Результати магістерської роботи доповідалися на таких конференціях:

– Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» [1];

За результатами магістерської кваліфікаційної роботи опубліковано тези доповідей у збірниках матеріалів конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз інформаційних війн

Поява інформаційних технологій та засобів масової комунікації суттєво вплинула на життя пересічних громадян, однак, ще більший вплив вони мали на проведення воєнних конфліктів. Для людей стало зрозуміло, що недостатньо перемагати на полі бою, адже це ніяк не допоможе підняти лояльність серед місцевого населення.

Історія боротьби з противником інформаційними методами почалася ще у Китаї у VI столітті до н.е. Відомий стратег Сунь-Цзи надав рекомендації з протидії супротивнику за допомогою психологічних та інформаційних засобів:

- намагатися послабити міць ворожих військ за рахунок перешкоджання нормальному постачанню і підтриманню порядку;
- послаблювати країну-жертву за рахунок дискредитації її історії, традицій, віри, лідерів, позитивних процесів тощо.

На думку Сунь-Цзи найбільш ефективним методом боротьби з ворогом є дезінформація: «Війна – це шлях омани. Якщо ти можеш що-небуть, показуй противнику, що не можеш; якщо ти користуєшся чим-небудь, показуй йому, нібито ти цим не користуєшся; хоч ти й близько від нього, показуй, нібито ти далеко; хоч ти й далеко від нього, показуй, що ти близько…» [2].

З розвитком технологій методи донесення «правильної» інформації змінювалися, однак ціль завжди залишалася однією – максимально дезінформувати ворога, шляхом введення його в оману та максимально підтримати власних громадян [2]. Так, в період Першої світової війни, були розповсюджені такі методи інформаційно-психологічного впливу:

- розповсюдження листівок (з використанням повітряних куль на території ворога було розкидано мільйони листівок з агітацією та закликами);
- розсилання листів (у Великій Британії розсилали пропаганду по домогосподарствам, а листи полонених з військових таборів, де

описували гарне ставлення до в’язнів, тиражували і розсилали по пошті);

– вплив на емоції (у пресі активно поширювали повідомлення про те, що німецькі солдати вчиняють «звірства», а союзники – мужні і героїчні воїни).

Очевидно, що кожен виток розвитку медіа відкривав нові можливості для пропаганди. Саме цим і скористалися націонал-соціалісти, щоб прийти до влади у Німеччині, а потім і вкласти мільйонам людей думку про «панівну расу».

Незважаючи на те, що поняття «інформаційної війни» вже давно введено в законодавство та є офіційною практикою багатьох країн, більшість користувачів все ще не приділяють увагу інформаційній гігієні. Так, люди залишають занадто багато відбитків в Інтернеті, що дозволяє супротивнику знаходити та використовувати потрібну інформацію на свою користь. Все це можливо завдяки OSINT.

Термін OSINT (Open Source Intelligence) означає розвідку на основі відкритих даних. Це включає в себе пошук, аналіз та використання загальнодоступної інформації для ухвалення рішень. Вперше збирати та аналізувати публічну інформацію розпочали під час Другої світової війни. Тоді, США створили службу моніторингу іноземного мовлення (FBMS) для аналізу інформації, яка поступала від іноземних організацій. Звісно, з появою інтернету пріоритети такого моніторингу змінилися з радіо на соціальні мережі. Так, за даними Агентства оборонної розвідки, приблизно 80% усіх розвідувальних даних отримуються безпосередньо з відкритих джерел [3].

З початком Революції Гідності українське суспільство зіштовхнулося з небаченою до того кампанією з дезінформації та дискредитації. Очевидно, що не всі були готові до інформаційно-психологічних операцій такого широкого масштабу, чим активно користувались російські медіа. Використовуючи широку мережу лояльних ресурсів, побудованих за часи незалежності, було розповсюджено надзвичайно велику кількість фейків, які не витримують жодної критики. В кінці лютого 2014 року у Криму почали з’являтися російські

військові, які почали захоплювати державні установи, штурмувати військові частини та встановлювати свою владу на півострові. В той же час, офіційна позиція Росії спростовувала будь-яку участь кадрових військових у цих діях. Саме в цей час почала створюватись відома міжнародна волонтерська група InformNapalm, яка мала на меті показати правду про присутність військових в Криму, в тому числі завдяки використанню OSINT інструментів [4]. Згодом, до українських громадян приєдналися волонтери з Німеччини, Чехії, Грузії. Очевидно, що після окупації Криму та початку гібридної війни на сході України спільнота ставала все більшою та виконувала все більше задач. Так, з початком Антитерористичної операції спеціалісти були залучені до прослуховування радіоканалів, складали карти блок-постів супротивника, використовували власних інсайдерів для отримання детальної інформації, допомагали військовим отримувати дані про знаходження ворожих підрозділів, адже лінія розмежування тоді була не встановлена. В той же самий час, офіційно, Росія всіляко заперечувала наявність своїх кадрових військових на території України та їх причетність до скоєння військових злочинів. Так, завдяки аналізу соціальних мереж, фото, відео та іншої інформації було виявлено докази присутності регулярних військ Російської Федерації на території України. Міжнародна організація Forensic Architecture створила сайт з інтерактивною мапою, де опубліковані фото та відео докази присутності російських військових на сході України (рис. 1.1) [5].

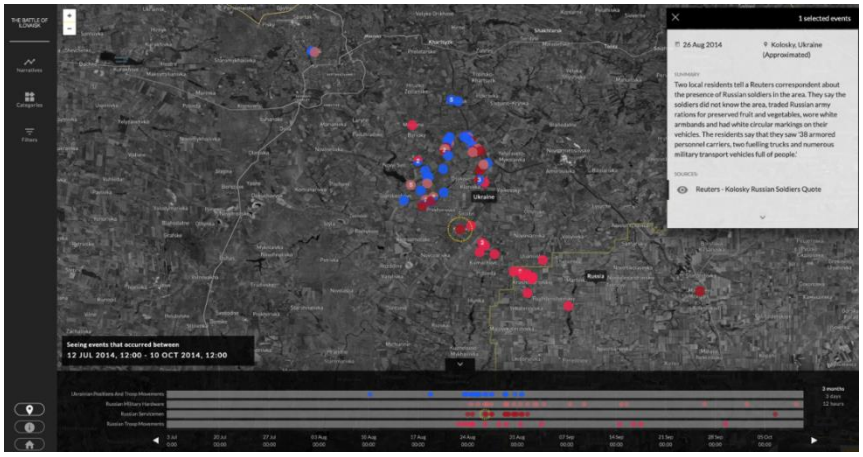


Рисунок 1.1 – Вигляд мапи з доказами присутності російських військ

На даному ресурсі опубліковані численні відеоролики з російськими полоненими солдатами, яких допитують українські військовослужбовці, а також відео та фото матеріали самих російських військовослужбовців, які вони публікували у соціальних мережах та на відповідних форумах в Інтернеті.

Значну роль OSINT спільнота відіграла також у розслідуванні катастрофи малайзійського Боїнга, яка сталася над Сніжним. Один із волонтерів команди InformNapalm дослідив профіль у соціальній мережі Vkontakte російського військовослужбовця автомобільного батальйону в/ч 83466 (147 Автобаза Генерального Штабу Міністерства оборони Російської Федерації) Дмітрія Зубова. За фото стало зрозуміло, що його підрозділ доставив ці зенітно-ракетні комплекси, один із яких, судячи з усього, і збив зрештою авіалайнер. Зі світлин стає очевидно, що солдат послідовно фіксував рух колони з «Буками». Але це було не останнє розслідування групи з приводу катастрофи малайзійського Боїнга. Активісти провели ще одне, в якому вони знайшли тягач Volvo, що перевозив ЗРК «Бук» безпосередньо територією, контрольованою бойовиками. Волонтери визначили місце перебування його на базі, що неподалік шахти «Ударник» в місті Сніжне.

З початком широкомасштабного вторгнення Росії 24 лютого 2022 року кількість роботи для OSINT спеціалістів значно збільшилась. Так, одним з найвідоміших випадків використання OSINT технологій є розкриття російських військових злочинів у Бучі [6, 7]. Після відступу з Київської області, росіяни намагалися приховати вбивства цивільного населення у Бучі. Американська компанія Maxar Technologies, що займається супутниковим зніманням Землі, надала знімки, на яких видно наслідки перебування російських військових на території міста. На даних знімках видно, що тіла цивільних знаходяться на тих же місцях, де їх знайшли українські військові. Таким чином, після співставлення даних знімків з наявними відеоматеріалами було вкотре доведено причетність російських військових до військових злочинів на території України (рис. 1.2) [8].

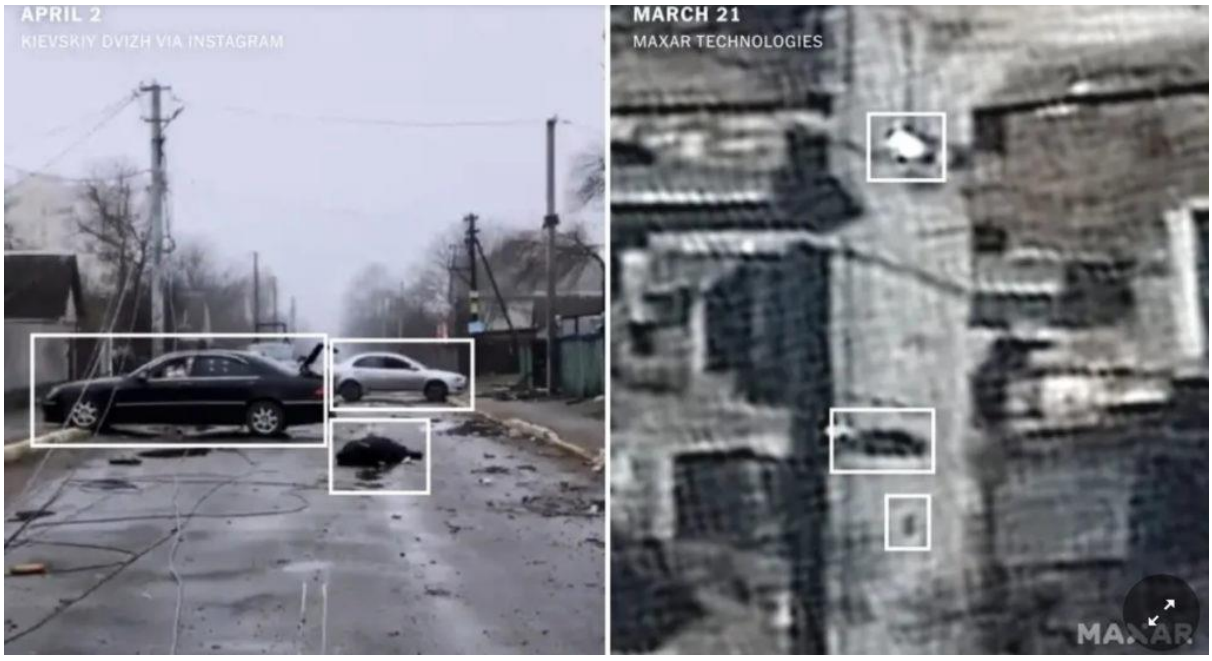


Рисунок 1.2 – Публікація New York Times щодо злочинів у Бучі

Для ідентифікації подібних інцидентів використовують опубліковані відео та фото у соціальних мережах, зіставлення геолокації, вигляд вулиць на GoogleMaps та супутникові знімки.

1.2 Методики атак в інформаційному просторі

З початком повномасштабного вторгнення Росії українське суспільство зіткнулося з великою кількістю інформаційно-психологічних операцій [9]. Інформаційно-психологічна операція – це дещо видозмінений переклад поняття Psychologica lOperations (PSYOPS). Якщо максимально спростити дане поняття, то основна мета проведення таких операцій – вплив на настрої груп у суспільстві. На наступному етапі поширення таких настроїв прямо позначиться на діях представників цього ж суспільства. Застосування інформаційно-психологічних операції не обмежується військовим часом.

На початку вторгнення дані операції мали на меті створити оптимальні умови для взяття під контроль територій, тобто зробити усе, щоб українці не чинили опір російським військам.

До елементів інформаційно-психологічних операцій відносяться дезінформація, пропаганда, перебільшення певної інформації або применшення іншої, диверсії в тилу, кібератаки. Часто певну інформацію запускають через майданчики пропаганди, щоб надалі вона поширювалася вже самими носіями, перетворюючи її у «вірусну» [10]. Інформація почута «сарафанним радіо» є найбільш небезпечною. Зазвичай, люди схильні довіряти оточуючим, особливо якщо одну й ту саму інформацію з певними висновками оточення ширить масово. Часте формулювання – «влада приховує правду», тоді як лише той чи інший Телеграм-канал або чат у Viber представляється як єдине надійне джерело інформації. Втім, елементарні знання з інформаційної гігієни могли б допомогти не стати багатьом українцям елементами розповсюдження неправдивої інформації. Більш глибокий аналіз «авторів» може показати синхронність публікацій, граматичні помилки, зокрема у назвах населених пунктів. Зазвичай, такі користувачі переважно є ботами, з автоматично згенерованими нікнеймами. Часто для прикриття використовуються елементи патріотичної символіки, наприклад прапор чи герб на фото у профілі соціальної мережі. При цьому, боти можуть підтримувати українську позицію, однак, іноді вказують на неточності у офіційно опублікованій інформації, тим самим ставлячи її під сумнів. Очевидною метою даних кампаній є деморалізація та паніка у суспільстві [11].

За часи холодної війни СРСР сильно просунувся у питаннях пропаганди та маніпулюванням людськими емоціями, а Росія покращувала та активно застосовувала дані методи. Далі буде наведено перелік основних з цих методів [12].

Велика брехня. Зазвичай, даний прийом націлений на західні країни та їх політиків. Так, за часи війни було помічено декілька доволі потужних інформаційних кампаній, які намагалися виправдати збройну агресію Росії проти України. Перша з них – інформація про надсекретні біолабораторії США на території України. Дана тема надзвичайно сильно поширювалася у російських медіа та інколи з’являлася на західних ресурсах. Однак, незважаючи на те, що перша інформація про це з’явилася на самому початку війни, Росія так і не надала

жодного вагомого доказу на рахунок своєї теорії. Восени Росія спробувала повторити свій «успіх», розпочавши нову кампанію з дезінформації. Так, міністр оборони по черзі телефонував своїм колегам з інших держав, повідомляючи, що Україна готує провокацією з брудною ядерною бомбою. Як і попереднього разу – дана кампанія не принесла жодного результату, а комісія МАГАТЕ, яку запросила Україна не виявила ніяких ознак підготування брудної бомби на об'єктах ядерної енергетики. Ще одним яскравим прикладом такого є повідомлення про те, що Польща, нібито готується до окупації Львівської області. Головною метою даних операцій є вплив на людей, які не вірять, що офіційні представники держави можуть розповідати відверту брехню.

«Очевидно, що…». Неправдиву інформацію подають так, ніби про неї всі знають і вона є очевидною для всіх. Основні тези таких повідомлень:

- санкції шкодять тим, хто їх запровадив, більше, ніж нам;
- вина за розв'язування війни проти України лежить на США, яким, начебто, вигідна війна, що призводить до послаблення Росії, а значить Китаю, а також до зростання цін на зерно і енергоресурси – продукцію традиційного американського експорту;
- звинувачення США у здійсненні тиску на Європу і роздмухуванні війни шляхом постачання зброї Україні.

Доволі часто у ЗМІ можна було побачити повідомлення, що агресія Росії проти України є, насправді, війною США і Заходу проти Росії руками України, якій постачають зброю замість спонукання до переговорів.

Маскування каналів у медіа. Не секрет, що Telegram за часи повноцінної війни став одним з основних методів отримання інформації для суспільства. Однак, готуючись до війни, Росія мала надзвичайно велику мережу каналів, що маскувались під українськи, при цьому поширювали російські наративи. Довіру маніпулятори завойовують псевдоукраїнським контентом. Так, 60% інформації може бути правдивою, а 40% - дезінформація та брехня. Для прикладу, в Україні й досі діє мережа з великих анонімних телеграм каналів, які були викриті СБУ

ще у 2021 році, як російські. Вони вдають, зо володіють інсайдерською інформацією, поширюють плітки про політику в Україні та світі.

«Гнилий оселедець». Мета даного прийому – руйнування репутації та довіри. Для реалізації використовують бездоказові звинувачення. Твердженням бракує підтверджень, проте підозри та сумніви все одно залишаються у підсвідомості та відіграють деструктивну роль. Наприклад, «влада в Україні повністю корумпована та неефективна», «волонтерські фонди наживаються на війні». Дискредитація легко чіпляється до об’єкта інформатаки та змушує його виправдовуватися.

Поділяй та володарюй. Не тільки в Україні, але й у світі пропаганда завжди шукає між людьми розбіжності, на яких можна зіграти. Таким чином, роздуваючи проблему можна поділити суспільство на ворогуючі табори, створивши конфлікт у суспільстві. Бідні – проти багатих, ліві – проти правих, україномовні – проти російськомовних. Тем для розколу надзвичайно багато. Надмірні емоції – одна з ознак, що зараз відбувається маніпуляція. Будь-яка суперечка повинна відбуватися з повагою до опонента. Деструктивні сперечання у соціальних мережах є однією з цілей ворожих інформаційно-психологічних атак, які мають на меті створити атмосферу недовіри у суспільстві.

Так, наприклад, всі українці пам’ятають одну з найбільш успішних російських інформаційно-психологічних кампаній [13]. Це мітки на будинках, дорогах, мостах та трасах. Основним завдання було посіяти паніку у суспільстві та створити атмосферу недовіри. В основному це спрацювало через те, що цивільні люди були необізнані у військовій сфері та реально припускали, що сучасні засоби ураження наводяться на ціль за рахунок міток на землі. Незважаючи на те, що офіційно влада спростувала будь-який вплив даних міток на цілі, що обираються для ураження, основна мета даної кампанії була досягнена, люди насправді почали сильно панікувати.

Однак, не завжди інформаційно-психологічні атаки націлені на суспільство в цілому, а можуть бути направлені проти певних осіб. Такі атаки називаються

таргетовані. Зазвичай, при їх створені активно використовуються методи соціальної інженерії.

Соціальною інженерією називають психологічні маніпуляції людиною, за допомогою яких можна отримати доступ до конфіденційної інформації про жертву [14]. Сьогодні існує чимало методів використання соціальної інженерії. В основі – маніпуляція людськими страхами, зацікавленістю або довірою. Жертвою соціальної інженерії можна стати як під час особистого спілкування, так і по телефону або через цифрові гаджети. Кіберзлочинці знайшли нові способи експлуатації людського актору – інстинктів цікавості й довіри,- які призводять до того, що люди з добрими намірами потрапляють у руки зловмисників. Це може статися у формі замаскованої URL-адреси, або, здавалося б, нешкідливого додатку в електронному листі. Але все, що потрібно, - це один клік, і негайно почнеться поширення шкідливої програми. Найпопулярніший серед численних сучасних інструментів соціальної інженерії – фішинг. Метою може бути заволодіння інформацією приватного характеру обманним шляхом. Попри те, що про нього широко відомо, успіх фішингу продовжує зростати через недостатню освіченість людей. Зловмисники можуть «маскуватися» під установи, яким довіряє людина. Наприклад, прикидаючись представниками оператора мобільного зв’язку або працівниками банку, вони можуть надсилати електронні листи з додатком або посиланням, за яким людина має ввести свої особисті дані [15].

Жертві також можуть додатково зателефонувати із проханням відкрити цей додаток або ж перейти за посиланням. Вважається, що таке «живе» спілкування додає ситуації чималої правдоподібності і зазвичай змушує людей відкривати вкладення.

На такий «гачок» потрапляли навіть досвідчені експерти з кібербезпеки. Нещодавно, наприклад, трьох українців заарештували за підозрою у викраденні – саме за допомогою фішингу – мільйонів номерів кредитних карток та атаках на більш ніж 100 американських компаній, що в результаті завдало збитків американським бізнесам у розмірі десятків мільйонів доларів [16].

З початком повномасштабної війни українці також почали масово отримувати фішингові листи, які маскувалися під офіційні. Приклади таких листів публікували СБУ, із закликом ігнорувати їх (рис. 1.3) [17].

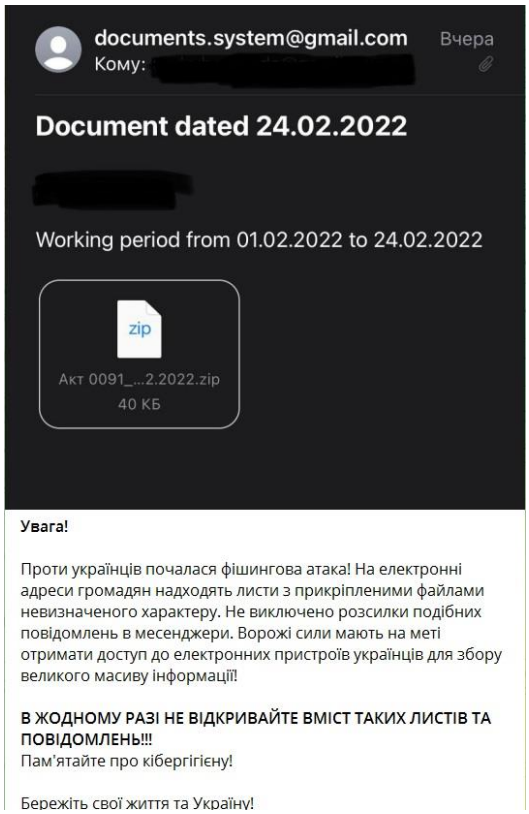


Рисунок 1.3 – Приклад фішингового листа

Так, у даному листі надсилається звіт, за робочий період за попередній місяць, щоб змусити користувача його переглянути.

Також, росіяни активно намагалися отримати будь-яку інформацію про військовослужбовців. Про це повідомляло МВС України (рис. 1.4).

Таким чином, ворог намагався отримати чутливу інформацію за рахунок емоційного навантаження на особу. Однак, не завжди таким атакам можна ефективно протидіяти, адже при правильному плануванні злочинці можуть мати надзвичайно багато інформації про потенційну жертву. Зазвичай, вони отримують їх за допомогою використання OSINT технологій. Для більш детального розуміння даного процесу у наступному розділі буде розглянуто основні способи отримання інформації з відкритих джерел.

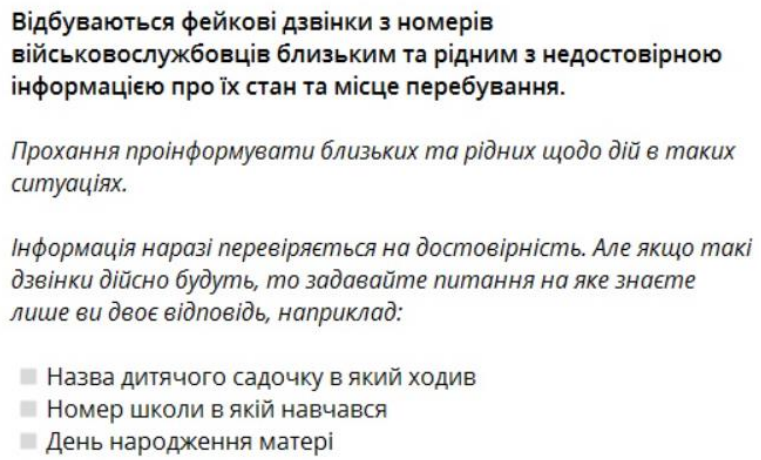


Рисунок 1.4 – Повідомлення МВС України щодо дзвінків родичам військовослужбовців

1.3 Джерела для отримання інформації про мішень

Збираючи інформацію та аналізуючи дані з відкритих джерел, зловмисник або пентестер має можливість сформувати повноцінний профіль жертви, визначити існуючі та потенційні вразливості. Цілеспрямовані кібератаки, як і військові атаки, починаються з попередньої розвідки, і перший та найбільш дієвий етап цифрової розвідки – це OSINT, пасивний збір розвідувальних даних без попередження цілі. Без активного залучення своєї жертви, зловмисник може використати отриману інформацію для побудови моделі загрози, розробки плану атаки або захисту.

Усі методи та засоби, що використовуються під час розвідки за відкритими джерелами можна поділити на дві категорії: пасивні та активні.

Пасивні методи дозволяють отримати загальну інформацію про досліджуваний об’єкт. Інформація збирається вручну з використанням спеціальних сервісів та інструментів, які значно спрощують збір, систематизацію та аналіз даних. Як приклад, можна навести програми, що дозволяють парсити сайти. Це дає змогу обробити велику кількість даних та подати їх у зручному для дослідника вигляді. Пасивною розвідкою можуть займатися всі, хто має комп’ютер та вільний доступ в Інтернет. До пасивних методів збору інформації відносять:

- збір інформації (в тому числі фото та відео матеріали) з відкритих пошукових систем;
- аналіз активності користувачів в соціальних мережах і блогах, на форумах або інших віртуальних платформах;
- пошук відкритих персональних даних користувачів у соціальних мережах або месенджерах;
- перегляд збережених копій сайтів у пошукових системах та інтернет-архівах;
- отримання геолокації даних з використанням загальнодоступних ресурсів, таких як GoogleMaps.

Під активними методами розуміються такі, де аналітик здійснює безпосередній вплив на досліджуваний об’єкт, використовує спеціалізовані засоби отримання даних або здійснює дії, які вимагають певних зусиль. До них відносять:

- збір даних на закритих ресурсах, доступ до яких обмежений та доступний лише за кошти;
- застосування спеціалізованих сервісів та програм, які активно впливають на досліджуваний об’єкт – наприклад, реєструються на сайті;
- використання сервісів, що сканують файли, програми, сайти на наявність шкідливого коду;
- створення підроблених веб-ресурсів, груп у соціальних мережах, каналів та чат-ботів у месенджерах, які збирають конфіденційні дані користувачів.

Під час виконання OSINT дослідження використання пасивних методів повинно плавно перетікати у використання активних методів, щоб зібрати більш детальну інформацію про досліджуваний об’єкт та мати повну картину. Далі буде розглянуто найбільш популярні OSINT інструменти.

Shodan – це пошуковий інструмент, який дозволяє знаходити пристрої, які підключені до мережі. Принцип роботи, насправді, є доволі простим. Даний

сервіс шукає та зберігає банери. В даному випадку банер це опис служби, яка використовується на пристрої. Приклад банеру можна отримати скориставшись наступною командою:

```
curl -I https://iq.vntu.edu.ua
```

Результат виконання команди наведено на рис. 1.5.

```
stepanhilko@Stepans-MacBook-Pro ~ % curl -I https://iq.vntu.edu.ua
HTTP/1.1 302 Found
Server: nginx
Date: Mon, 24 Oct 2022 13:38:50 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
Set-Cookie: PHPSESSID=atl0ln8th52hrnu1omv0g4lj51; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: https://jetiq.vntu.edu.ua
X-Frame-Options: SAMEORIGIN
```

Рисунок 1.5 – Приклад банеру, який використовує Shodan

З прикладу видно, що виконання команди повертає відповідь на запит. В ній знаходиться назва протоколу (HTTP) та код статусу 302. По кодам можна зрозуміти, як сервер реагує на той чи інший запит:

- 200 – все добре, запит було успішно оброблено;
- 300 – перенаправлення на іншу сторінку;
- 400 – проблема з запитом;
- 500 – проблема з сервером.

Після цього можна побачити безпосередньо сам заголовок. В ньому зберігається різноманітна технічна інформація. Вона, очевидно, для кожного протоколу різна. Також в банері може міститися багато детальної інформації про сам пристрій, що, зазвичай, може бути корисно. Також Shodan збирає метадані пристроїв, наприклад: геолокацію, операційну систему тощо.

Для прикладу, спробуємо знайти всі tp-link, які під’єднані до мережі. Для цього, в пошуку треба ввести «tp-link» та натиснути пошук. Результати наведено на рис. 1.6.

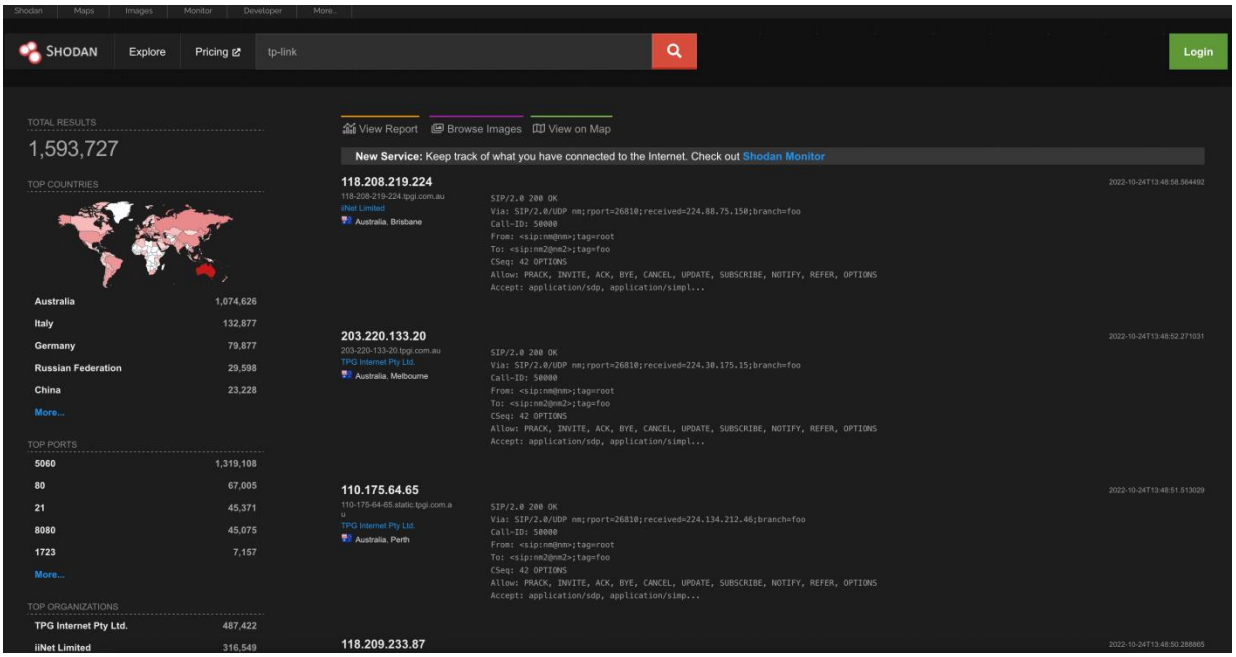


Рисунок 1.6 – Результат пошуку за запитом tp-link

Однак, через велику кількість результатів, дуже важко знайти потрібну, тому доцільно скористатися фільтрами. Найбільш популярні фільтри Shodan:

- country: - пошук по обраній країні, наприклад country:UA;
- city: - пошук по обраному місту, наприклад city:Kyiv;
- os: - пошук по операційній системі, наприклад os:linux;
- geo: - пошук по геолокації, де необхідно вказати координати, наприклад geo:”45.4545, 55.5555”;
- port: - пошук по обраному порту, наприклад port:22;
- hostname: - пошук по доменній зоні, наприклад hostname:.ua;
- net: - пошук по заданому мережевому діапазоні, наприклад net: 190.74.41.51/24;
- product: - пошук по назві утиліти, яка повернула банер, наприклад product:openssh.

Тепер, можна використати дані фільтри і отримати більш детальні результати пошуку. Наприклад, переглянемо список веб-камер у місті Київ, які використовують порт 8080. Для цього потрібно скористатися наступною командою:

```
Webcam country:UA city:Kyiv port:8080
```

Результат виконання команди наведено на рис. 1.7.

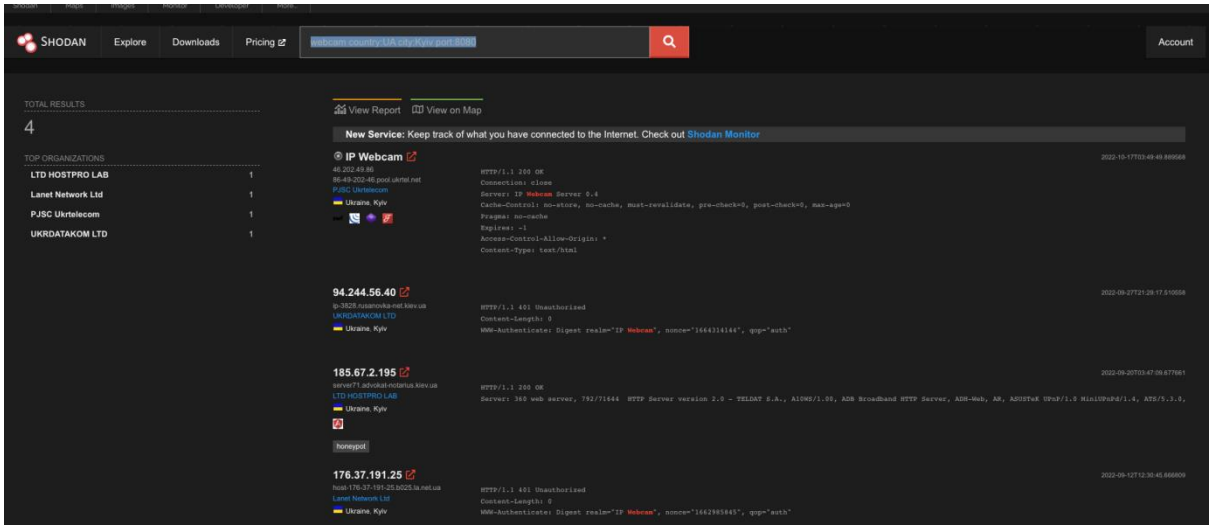


Рисунок 1.7 – Результат виконання пошуку Shodan з фільтрами

Як видно з результатів, було знайдено 4 веб-камери, які знаходяться у Києві та використовують порт 8080. Таким чином, можна отримати доступ до веб-камери потенційної жертви, адже авторизація на таких веб-камерах, зазвичай, доволі легко ламається методом грубої сили.

OSINT Framework – це веб-ресурс, який містить у собі посилання на велику кількість інших ресурсів, які можна використовувати для пошуку необхідної інформації про мішень. Зовнішній вигляд ресурсу наведено на рис. 1.8.

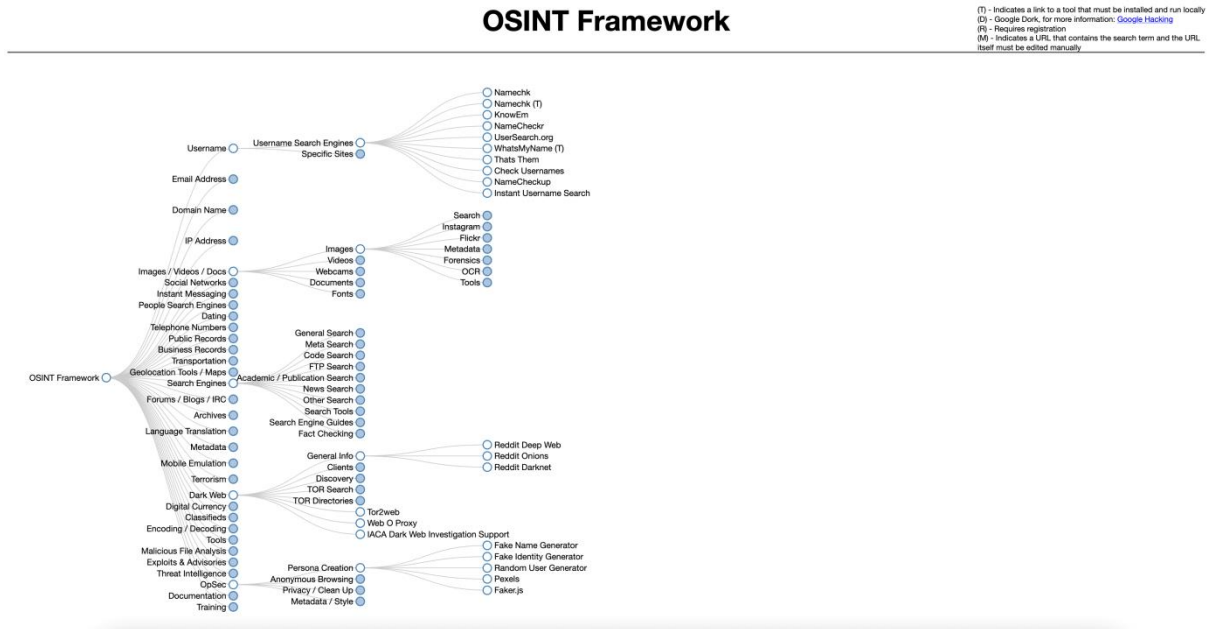


Рисунок 1.8 – Вигляд сайту OSINT Framework

Як видно з ілюстрації, на сайті наявна велика кількість інструментів, відсортованих по категоріям.

Так, наприклад, можна спробувати знайти користувача за логіном та перевірити, якими соціальними мережами він потенційно може користуватися. Зробити це можна на сайті namecheckr.com, для цього потрібно просто ввести логін цілі та дочекатися результатів (рис. 1.9).

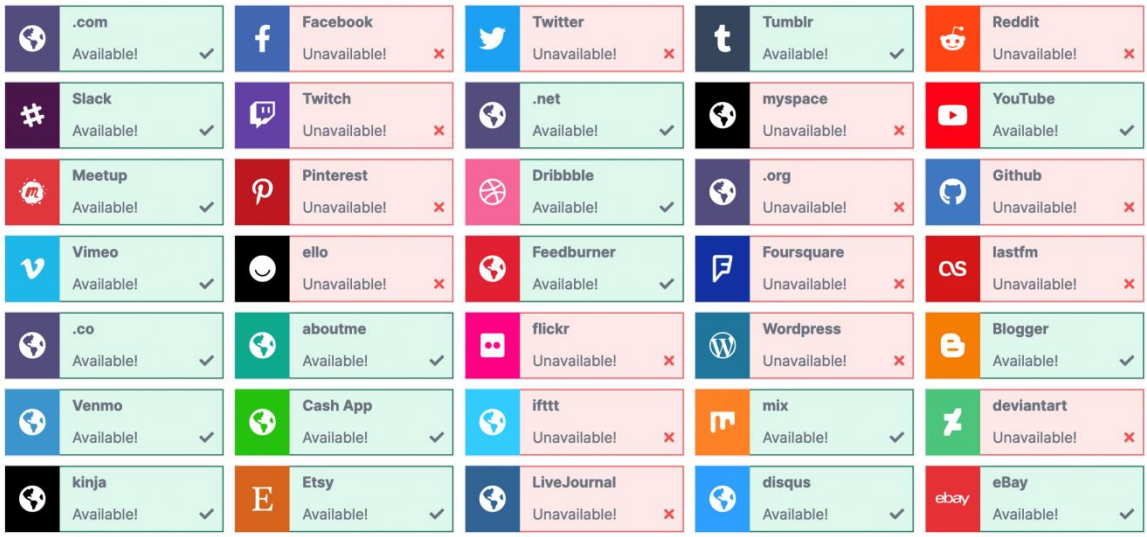


Рисунок 1.9 – Результат пошуку за логіном

Таких ресурсів багато і кожен з них використовує свій власний список соціальних мереж, по яким відбувається пошук. Очевидно, що можливості використання OSINT Framework надзвичайно широкі і кожен інструмент потрібно підбирати під конкретну задачу, яка стоїть перед дослідником.

Також, важливими інструментами відкритої розвідки є Nmap та Recon-ng. Nmap дозволяє вказати IP-адресу та визначити, які хости доступні, які браундмауери або операційні системи вони використовують. Приклад використання Nmap наведено на рис. 1.10.

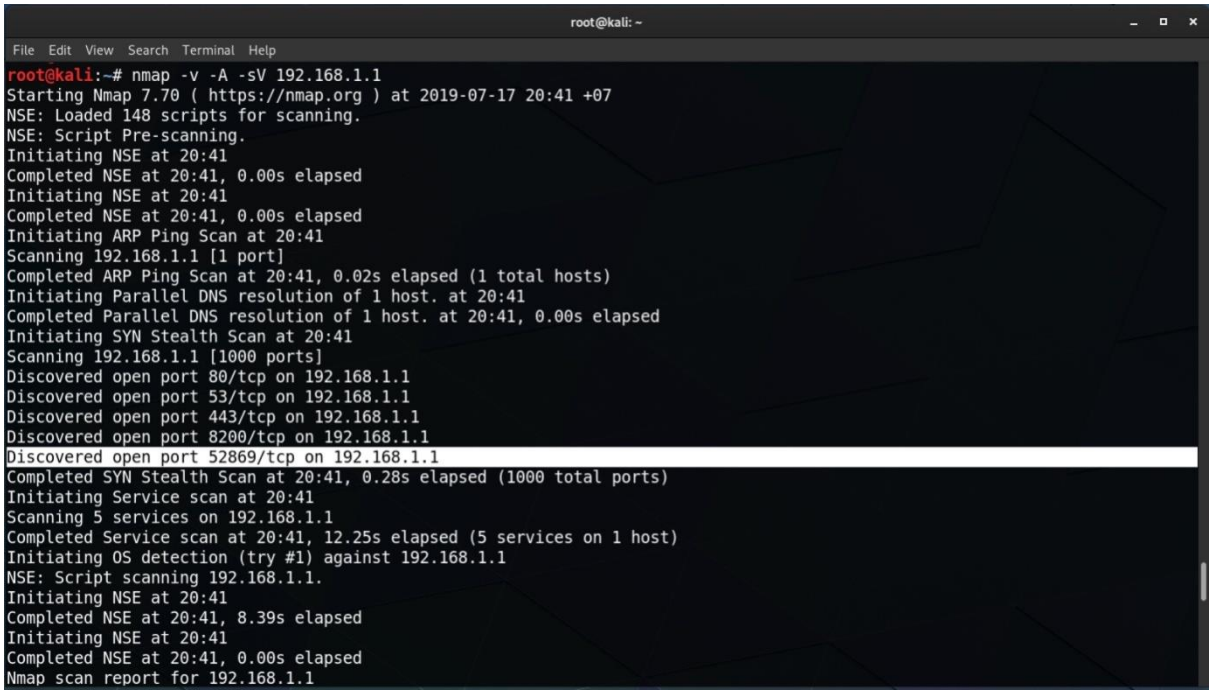


Рисунок 1.10 – Результат сканування за допомогою Nmap

Google Dorks або Google Hacking – техніка, яка використовується ЗМІ, правоохоронними органами, інженерами з безпеки та OSINT спеціалістами. Вона дозволяє створювати запити, які розкривають приховану інформацію, що знаходиться на загальнодоступних серверах. Взагалі, Google має надзвичайно велику кількість фільтрів, про які не здогадуються звичайні користувачі. Саме це і дозволяє здійснювати пошук більш детально, фільтруючи результати пошуку за URL, розширенням файлу та його вмістом, заголовком сторінки тощо. Далі буде наведено результати пошуку з використанням фільтрів.

Наприклад, спробуємо знайти список файлів з розширенням PDF, які мають у тексті прізвище Лужецький. Для цього потрібно скористатися наступними фільтрами пошуку:

inurl:vntu filetype: pdf + Лужецький

Результат виконання пошуку наведено на рис. 1.11.

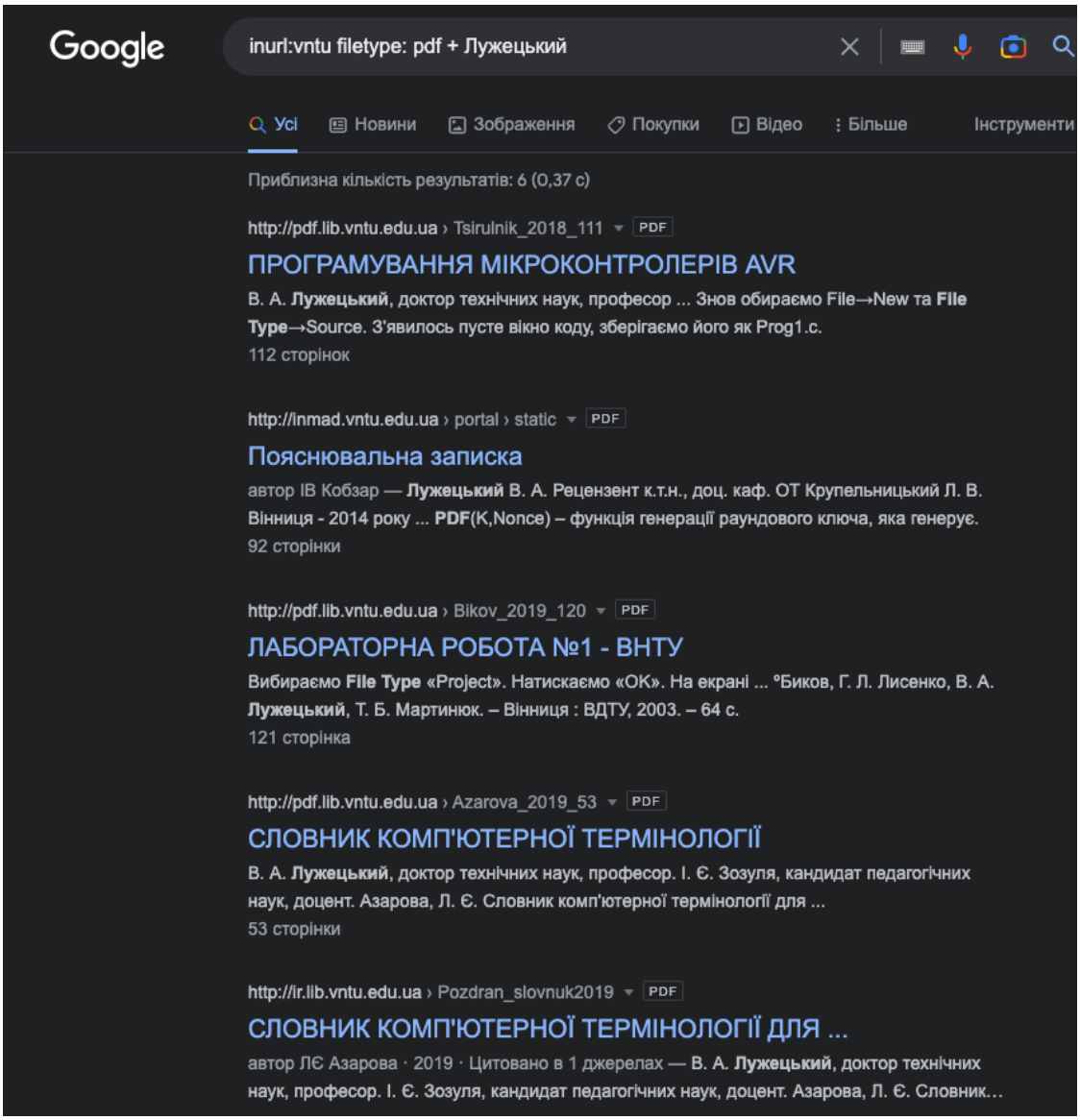


Рисунок 1.11 – Результати пошуку в Google з фільтрами

Так, з результату видно, що було знайдено перелік PDF файлів, які відповідають результатам пошуку. Таким чином можна спробувати знайти паролі, які були збережені у відповідних файлах. Так, наприклад, можна використати веб-ресурс Google Hacking Tests [18]. Він дозволяє здійснювати пошук за найбільш популярними запитами до Google Dorks, автоматично формуючи запит за заданими критеріями.

Перелік OSINT технік, список інформації, які з їх допомогою можна отримати та перелік відомостей наведено на рис. 1.12.

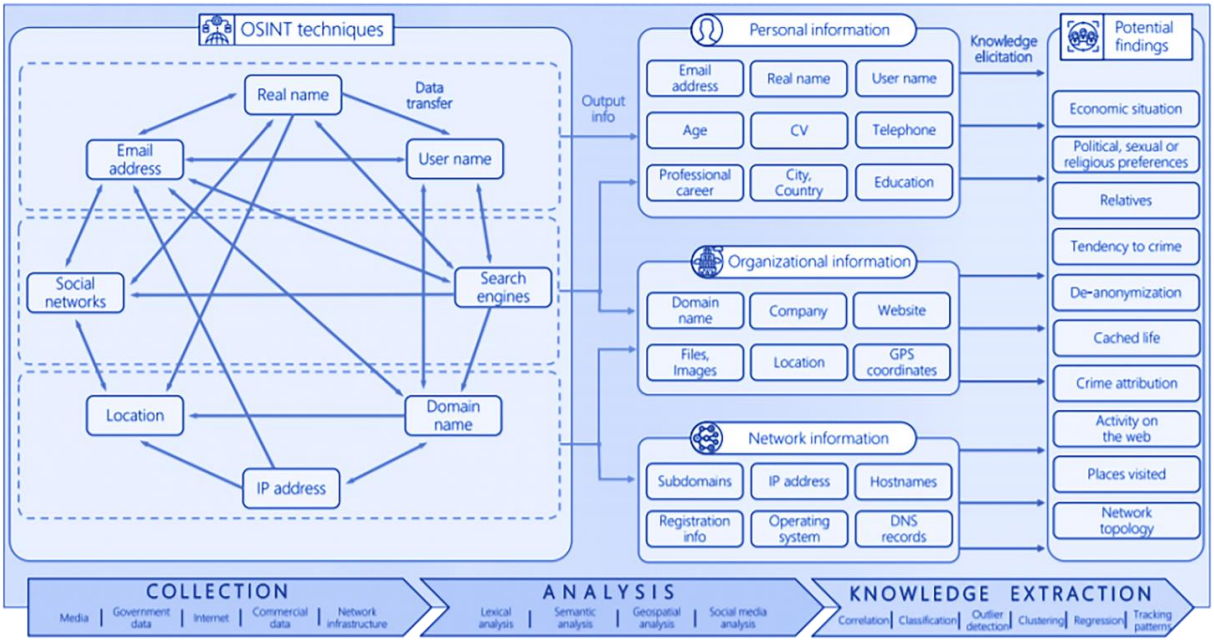


FIGURE 2. Principal OSINT workflows and derived intelligence.

Рисунок 1.12 – Опис основних OSINT-технік

Так, з рисунку видно, що первинною інформацією може бути: справжнє ім’я, логін, поштова адреса, соціальні мережі, місце знаходження, доменне ім’я, IP-адреса. З даного переліку первинної інформації можна отримати перелік інформації з другої колонки, наприклад: дані про досвід роботи, навчання, мобільний телефон, резюме, адреса сайту, картинки та ін. З даних знань формується розуміння про об’єкт OSINT дослідження, перелік можливих знань знаходиться у третій колонці. Це може бути економічна ситуація, політичні уподобання, родичі, залученість до кримінальних злочинів та ін.

1.4 Формалізація вимог та постановка задачі

Проаналізувавши доступні інструменти, можна зробити висновок, що вони дають змогу знайти надзвичайно велику кількість інформації про мішень. Однак, більшість з них використовуються з конкретною метою та мають свою вузьку спеціалізацію. Також, більшість з них надають дані в незручних форматах для користувача, тому виникає необхідність створити свій власний сервіс, який буде об’єднувати декілька інструментів в одному та мати змогу експортувати дані в зручному для користувача форматі. Розробка включатиме декілька етапів:

1) Аналіз сценаріїв атак та побудова моделей інформації, яку необхідно збирати на кожному етапі.
2) Обрання необхідних інструментів, які будуть використовуватися для збору інформації в залежності від потреб дослідника.
3) Створення веб-інтерфейсу, який дозволить користувачу взаємодіяти з пошуком та зберігати його результати.
4) Створення backend-частини, яка матиме підключення до бази даних. Це дозволить обробляти запити користувачів у фоновому режимі, зберігати необхідні дані та експортувати результати в необхідному для користувачів форматі.

Отже, що існуючі програми часто не включають в себе всю необхідну інформацію, або не надають її у зручному для користування вигляді, що сповільнює та утруднює розгортання кібератаки. Поставлено завдання на розробку методу та засобу збору інформації про мішень, який би враховував етапи проведення інформаційної операції та надавав саме ту інформацію, яка потрібна на цьому етапі.

.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

2.1 Аналіз сценаріїв атак

Очевидно, що наявність інформації про людину не дає ніяких практичних переваг, вона стає необхідною під час проведення інформаційних операцій під час інформаційної війни.

Так, наприклад, розглянемо модель класичної фішингової атаки з використанням електронної пошти. Її алгоритм наведено на рис. 2.1 [19].

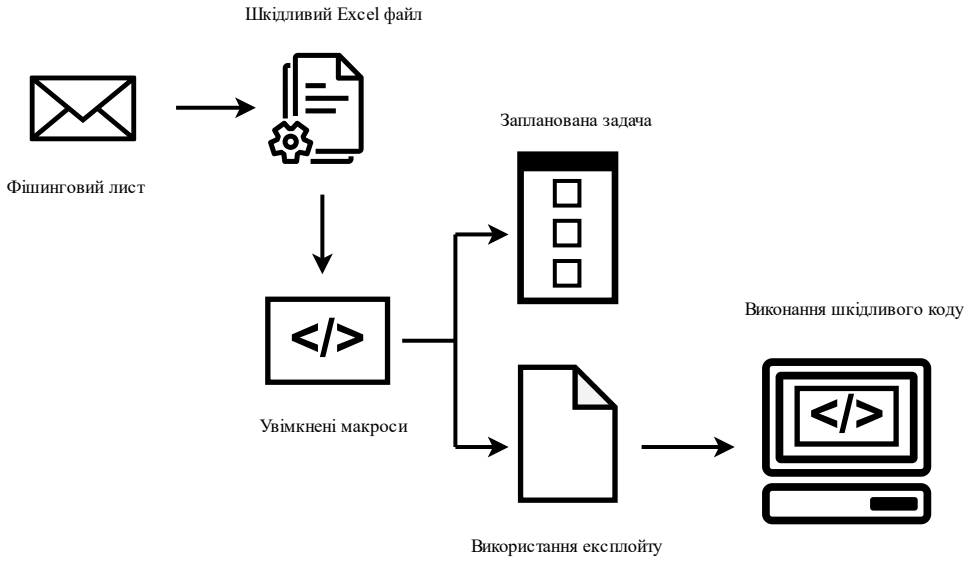


Рисунок 2.1 – Алгоритм фішингової атаки з використанням електронної пошти

Оскільки, система дозволятиме отримувати електронну пошту цілі дана атака є одним із потенційних варіантів використання системи. Так, користувач отримує фішинговий лист, до якого прикріпляється Excel документ. Даний лист може містити будь-яку інформацію, наприклад бухгалтерські виписки, звіт з роботи за певний період часу та ін. Припускається, що користувач завантажить файл, який було прикріплено до листа. Таким чином, файл потрапляє на комп’ютер цілі. Зазвичай, пересічні користувачі не знають, що за допомогою Excel їх комп’ютер можна заразити шкідливим програмним забезпеченням.

Для цього використовуються XML-макроси. Дана технологія підтримується усіма версіями Microsoft Office Excel до Office 2016. Для створення XML-макроса в Excel необхідно виконати певні налаштування у самому документі. Для цього необхідно натиснути на комірку та вибрати «Вставити», після чого відкриється відповідне діалогове вікно, його наведено на рис. 2.2.

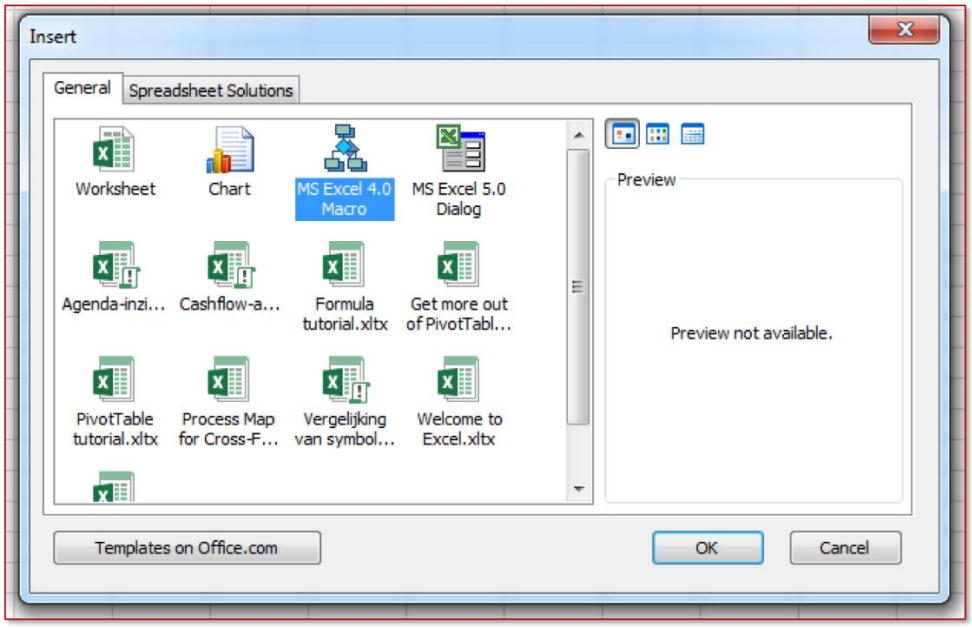


Рисунок 2.2 – Вигляд вікна для активації макросу

У цьому вікні необхідно обрати MS Excel 4.0 Macro. Після цього, в Excel файлі з’явиться сторінка Macro1, де можна писати свої власні макроси. У будь-якій комірці даного файлу можна запустити необхідний процес. Для автоматичного запуску макроса першу комірку необхідно перейменувати в «Auto_open». Так, після цього можна викликати будь-які додатки, що наявні у Windows (рис. 2.3).

Так, на рисунку наведено приклад виклику калькулятора. Після виконання даного макросу – відкриється стандартний калькулятор операційної системи Windows. Також, сторінку з макросами можна приховати, натиснувши відповідну кнопку у контекстному меню. Окрім можливості запуска процесів, XML-макроси вміють звертатися до Win32 API за допомогою функцій REGISTER і CALL.

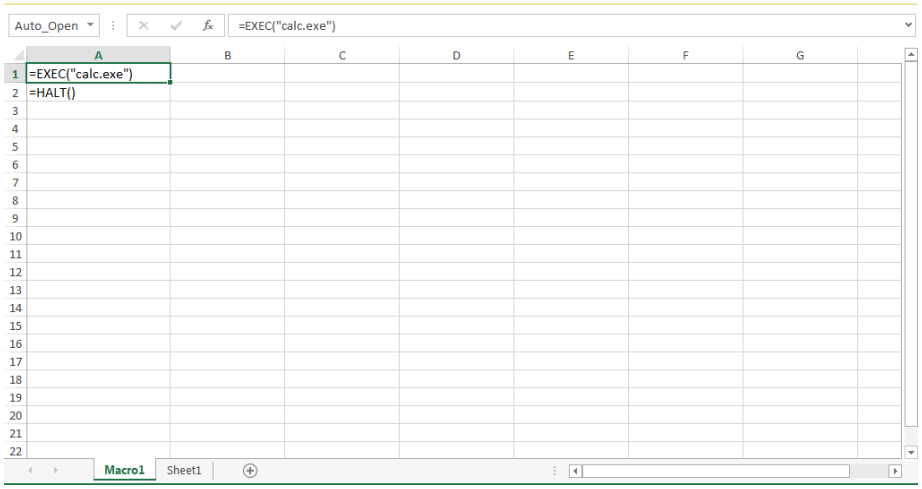


Рисунок 2.3 – Приклад виклику калькулятора

Далі наведено приклад PoC-коду макроса, який вбудовує шел-код (рис. 2.4).

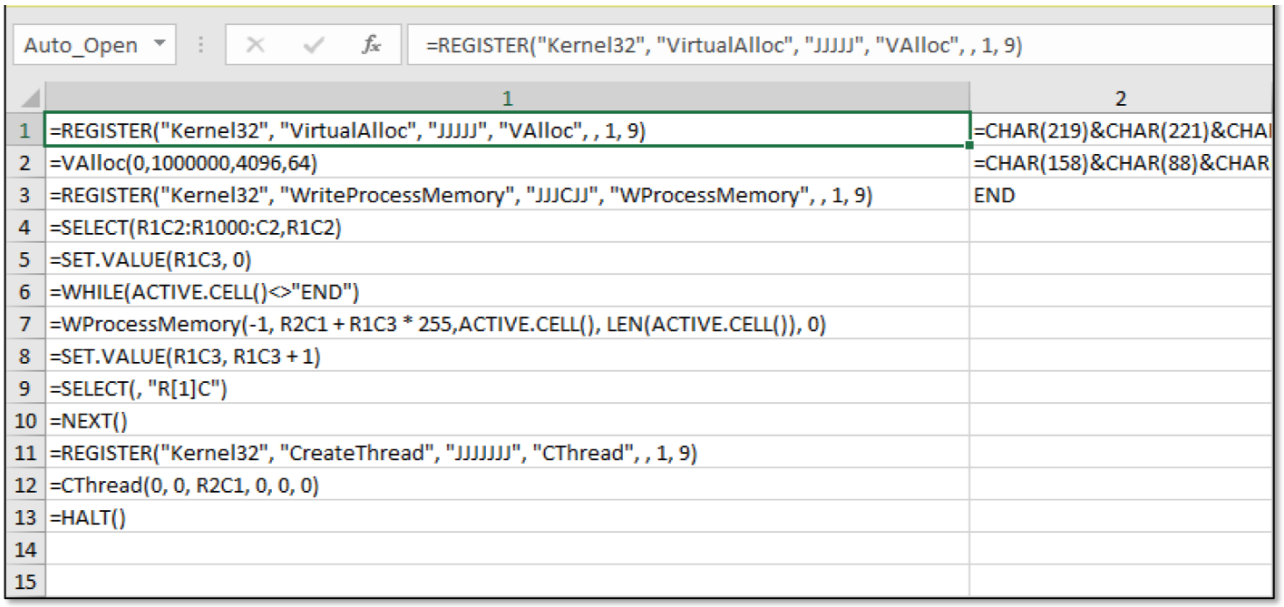


Рисунок 2.4 – Вбудовування шел-коду за допомогою макроса

Таким чином, можна вбудувати буд-який код за допомогою Excel файлу та змусити комп’ютер цілі виконувати необхідні дії.

Суттєвим недоліком є те, що користувач повинен дозволити виконання макросів, що значно ускладнює задачу. Адже обізнані користувачі ніколи не дозволять стороннім файлам запускати макроси на своєму комп’ютері. Тому, далі буде наведено приклад експлуатації вразливості, яка дозволяє виконувати

будь-який код за допомогою програм Microsoft Office та не потребує будь-яких взаємодій з боку користувача.

Дана вразливість була знайдена у 2021 році в браузерному рушію від Microsoft – MSHTML. Зловмисники створювали офісний документ, який містить спеціальний шкідливий об’єкт. Якщо жертва відкривала файл, додаток Microsoft Office завантажував і виконував шкідливий скрипт. Приклад алгоритму з використанням такої атаки наведено на рис. 2.5.

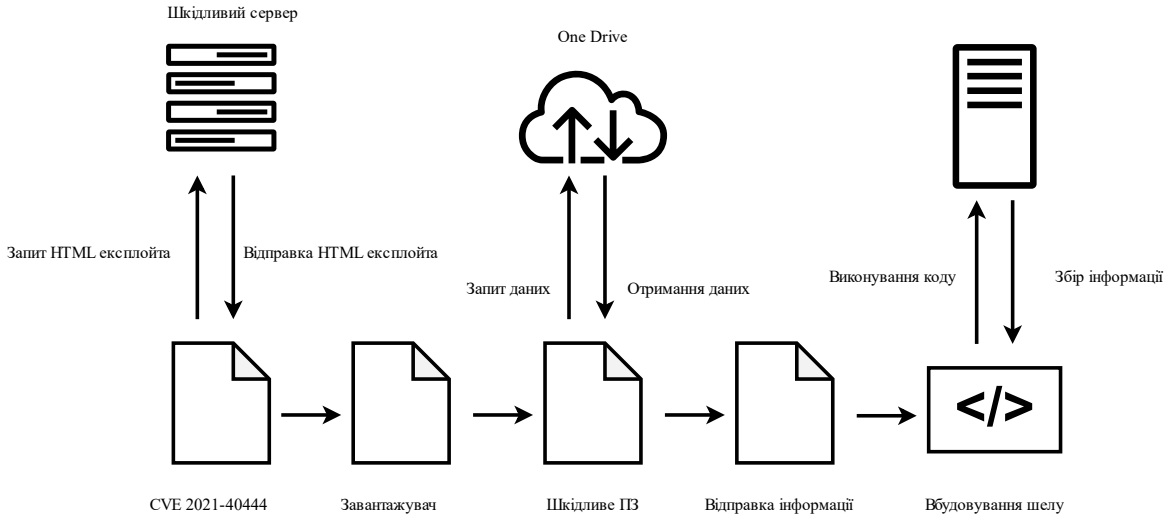


Рисунок 2.5– Використання CVE 2021-40444

Ця вразливість має назву CVE 2021-40444. На електронну пошту потенційної жертви надходить фішинговий лист, який має спонукати користувача завантажити документ собі на комп’ютер.

Для експлуатації вразливості зловмисники вбудовують у документ Microsoft Office спеціальний об’єкт, що містить посилання на шкідливий скрипт. При відкритті цього документу Microsoft Office завантажить шкідливий скрипт по вбудованому посиланню і запустить його за допомогою MSHTML. Після цього скрипт може використовувати елементи керування ActiveX для виконання шкідливих дій над комп’ютером цілі.

Таким чином, після відкриття виконається шкідливий код. З рисунку видно, що це може бути будь-яка послідовність дій. Після того, як за допомогою завантажувача було встановлено шкідливе програмне забезпечення на комп’ютер цілі воно може збирати інформацію та запускати інші скрипти. Так,

наприклад, можна викачати інформацію з хмарного сховища OneDrive, відправити її на сервери зловмисників та вбудувати шкідливий шел-код, який продовжить слідкувати за комп'ютером жертви, збирати інформацію та відправляти її до зловмисників.

З даних пояснень стає зрозуміло, що головна складність – це не можливість виконати дії на комп'ютері користувача, а правильне спонукання користувача до виконання необхідних дій, що передують атаці.

Загальну модель атаки наведено на рис. 2.6.

Рисунок 2.6 – Модель проведення атаки

На кожному з цих етапів необхідно використовувати різні дані про мішень для проведення інформаційної атаки.

2.2 Розробка методу збирання інформації на кожному етапі інформаційної операції

З попереднього опису можна зрозуміти, що фішинг є одним із найпопулярніших і найпростіших способів проведення інформаційних операцій. Багато хто з користувачів регулярно отримує фішингові повідомлення, однак, якщо вони і проходять спам-фільтр електронних адрес чи телефонних крижок, то, зазвичай, не вражають своєю ефективністю. Це відбувається тому, що інформаційна операція націлена на широку групу осіб. Зловмисники намагаються охопити найбільш широкий пласт населення, щоб зібрати якомога більше інформації. Таким чином, надаючи перевагу кількості втрачається якість повідомлень. Однак, у контексті інформаційної війни – це зовсім неефективний підхід. Зазвичай, є необхідність отримати доступ до даних однієї особи або вузької групи осіб. Для цього можна використати атаки, які були описані раніше, проте необхідно змусити користувача відкрити необхідне посилання або ж завантажити певний файл та запустити його у себе на комп’ютері.

Для створення ефективних повідомлень необхідно мати інформацію про людину, на яку здійснюватиметься атака. Саме таку інформацію дозволяє отримати розроблена система, а саме:

- номер телефону;
- електронна адреса;
- паспортні дані;
- профілі у соціальних мережах;
- залученість особи до судових засідань;
- пошук паролів по логіну у соціальних мережах або електронній адресі;
- фото з соціальних мереж.

Зрозуміло, що при наявності доступу до будь-якої з основних соціальних мереж цілі задача створення ефективного фішингового листа значно спрощується, адже там наявна надзвичайно велика кількість приватної інформації. Таким чином, створені листи з використанням даної інформації

матимуть вищий рівень довіри, спонукаючи користувача до дій. Однак, у більшості випадків отримати доступ до соціальних мереж не так легко, тому необхідно шукати інші варіанти.

Так, наприклад, можна використовувати інформацію, що наявна у відкритому доступі. Наприклад, якщо у вас є посилання на профіль у соціальній мережі LinkedIn ви можете скористатися наступним алгоритмом пошуку інформації про мішень (рис. 2.7).

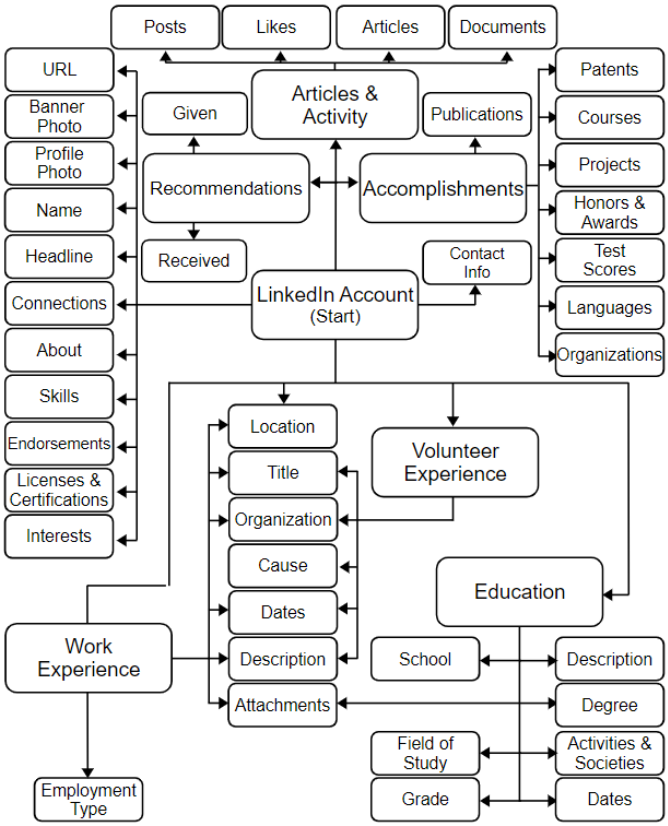


Рисунок 2.7 – Модель пошуку інформації про мішень за допомогою LinkedIn

Ця модель створена групою OSINT Dojo [20]. Це група ентузіастів, які створили сайт для поглиблення у сферу OSINT з безліччю корисних посилань на ресурси та програми, що дозволяють підвищити свої OSINT навички. На даній діаграмі визначається перелік інформації, яку можна отримати про людину з одного лише профіля LinkedIn. Варто відзначити, що LinkedIn – це соціальна

мережа для професіоналів, яку використовують для пошуку роботи, тому інформація в ній максимально достовірна, а користувачі мережі самі її надають.

Так, діаграма показує, що почати аналіз сторінки необхідно почати з того, щоб зібрати досвід роботи, досвід волонтерства та інформацію про навчання цілі. З цього переліку можна отримати більш детальну інформацію, наприклад, організації, де працював, тип працевлаштування, школа, університет, спеціальність, ступінь, оцінки, відзнаки та дипломи.

Далі можна переходити до аналізу контактів користувача. Зазвичай, користувачі LinkedIn мають багато контактів, тому що їм часто пишуть рекрутери з пропозиціями роботи. Однак, діаграма пропонує переглядати список рекомендацій, які є на сторінці кожного користувача. Так, переглянувши список людей, які підтвердили ті чи інші навички ми можемо зрозуміти, з ким потенційна мішень насправді працювала, щоб використати цю інформацію у майбутньому.

Так, наприклад, можна скористатися інформацією про попереднє місце роботи мішені. Гроші є однією з основних слабкостей, тому доцільно тиснути саме на цю точку. Скориставшись пошуком по співробітникам можна знайти ім’я бухгалтерів фірми та їх контактними даними. Після цього, створити формальний лист, з урахуванням імені цілі, імені бухгалтера та написати текст про те, що компанія не виплатила співробітнику необхідну суму коштів. Щоб її отримати необхідно заповнити прикріплений документ та відправити її на вказану поштову адресу. Так, за рахунок того, що лист буде сформований за формальними вимогами, максимально подібним до справжніх листів, та міститиме у собі мотивацію до дії, можна значно підвищити шанси того, що користувач насправді взаємодіятиме з листом та завантажить необхідний документ Microsoft Office.

Модель з видами інформації, що потрібні на кожному етапі атаки, наведено на рис. 2.8.

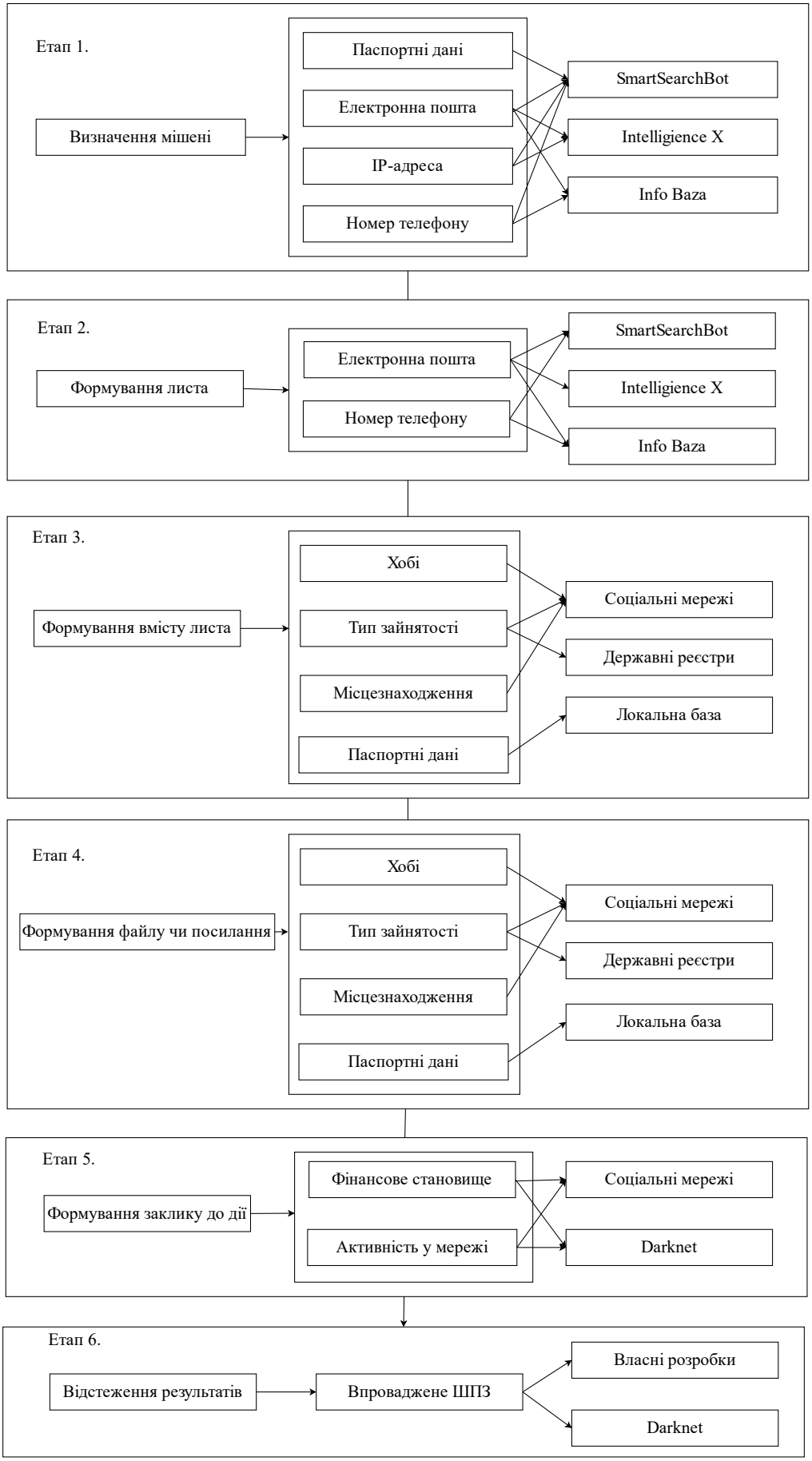


Рисунок 2.8 – Модель проведення інформаційної операції з урахуванням видів та джерел інформації на кожному етапі

Так, на першому етапі необхідно визначити мішень. Для цього потрібні наступні дані: паспортні дані, електронна пошта, IP-адреса, номер телефону. Дану інформацію можна отримати з наступних баз даних або сервісів: SmartSearchBot, IntelligienceX, InfoBaza.

На другому етапі необхідно почати формувати лист. Оскільки лист або СМС-повідомлення доставляється або на електронну пошту або на номер телефону доцільно скористатися тими самими базами даних.

На третьому етапі необхідно сформувати вміст листа, який зачепить потенційну мішень. Так, можна проаналізувати хобі, тип зайнятості, місцезнаходження та паспортні дані. Використавши цю інформацію можна створити таргетований лист. Дану інформацію можна отримати у соціальних мережах, державних реєстрах або у локальній базі системи, оскільки дані з попереднього етапу зберігаються.

На четвертому етапі можна використати ті самі дані, адже файл чи посилання так само повинні бути дотичними до потенційної цілі.

На п’ятому етапі необхідно сформувати заклик до дії. Для цього можна скористатися таким тригером, як фінансове становище або відслідкувати активність цілі у мережі. Це можуть бути злиті історії покупок, дані банківських карток, пошукові запити тощо.

На останньому, шостому етапі, необхідно відслідковувати результати виконання атаки. Якщо за мету ставиться впровадження шкідливого програмного забезпечення, то його можна отримати у Darknet або створити власне.

2.3 Розробка структури програмного засобу збирання інформації про мішень

Програмний засіб складається з процесів, які є незалежними частинами, що дає змогу виявляти проблеми на ранніх стадіях проектування та швидко їх усунути. Складові інформаційної технології представлено на рис. 2.9.

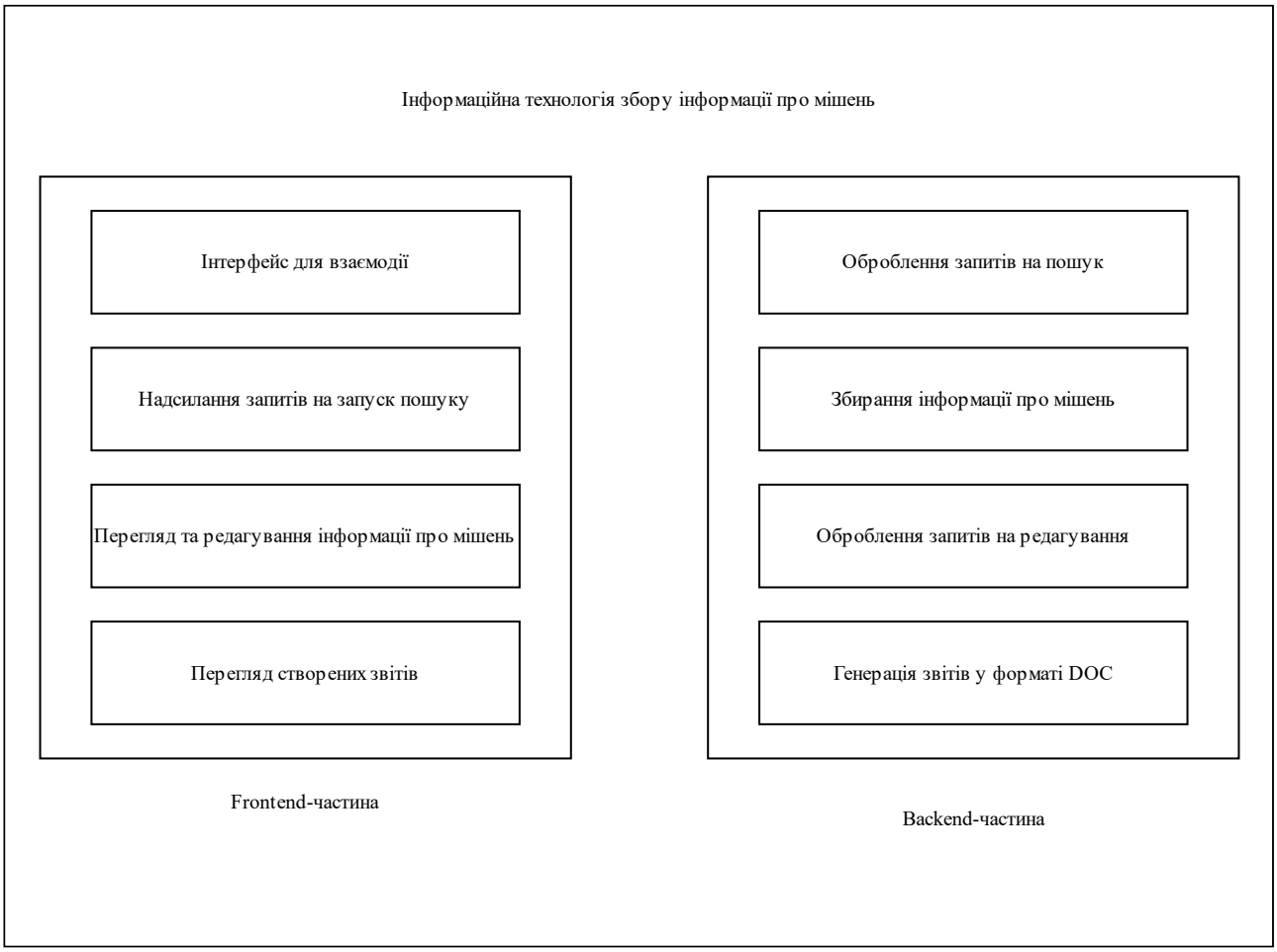


Рисунок 2.9 – Складові програмного засобузбирання інформації про мішень

Програмний засіб поділяється на дві складові:

– Frontend-частина – дозволяє користувачу зручно взаємодіяти з інформаційною технологією, надсилати запит на запуск пошуку інформації про нову мішень, переглядати та редагувати наявну інформацію про мішень, переглядання згенерованих звітів у форматі DOC;
– Backend-частина – обробляє запити користувачів, безпосередньо збирає інформацію про мішень, обробляє запити на редагування інформації, генерує звіти у форматі PDF.

Перше, що бачить користувач, коли відвідує відповідний веб-ресурс – це Frontend-частина. Це все, що браузер виводить на екран користувача. У даному випадку, необхідно створити базовий функціонал, який дозволяє зручно

взаємодіяти користувачу з інформаційною технологією. Також, необхідно відображати «цілі», які наявні у системи та всю інформацію, що їх стосується, відправляти запити до Backend-частини та відображати звіти, згенеровані системою. Оскільки інформаційна система, що створюється, буде реалізована у формі веб-ресурсу необхідно мати розуміння того, як працюють веб-ресурси. Схему типової взаємодії наведено на рис. 2.10.

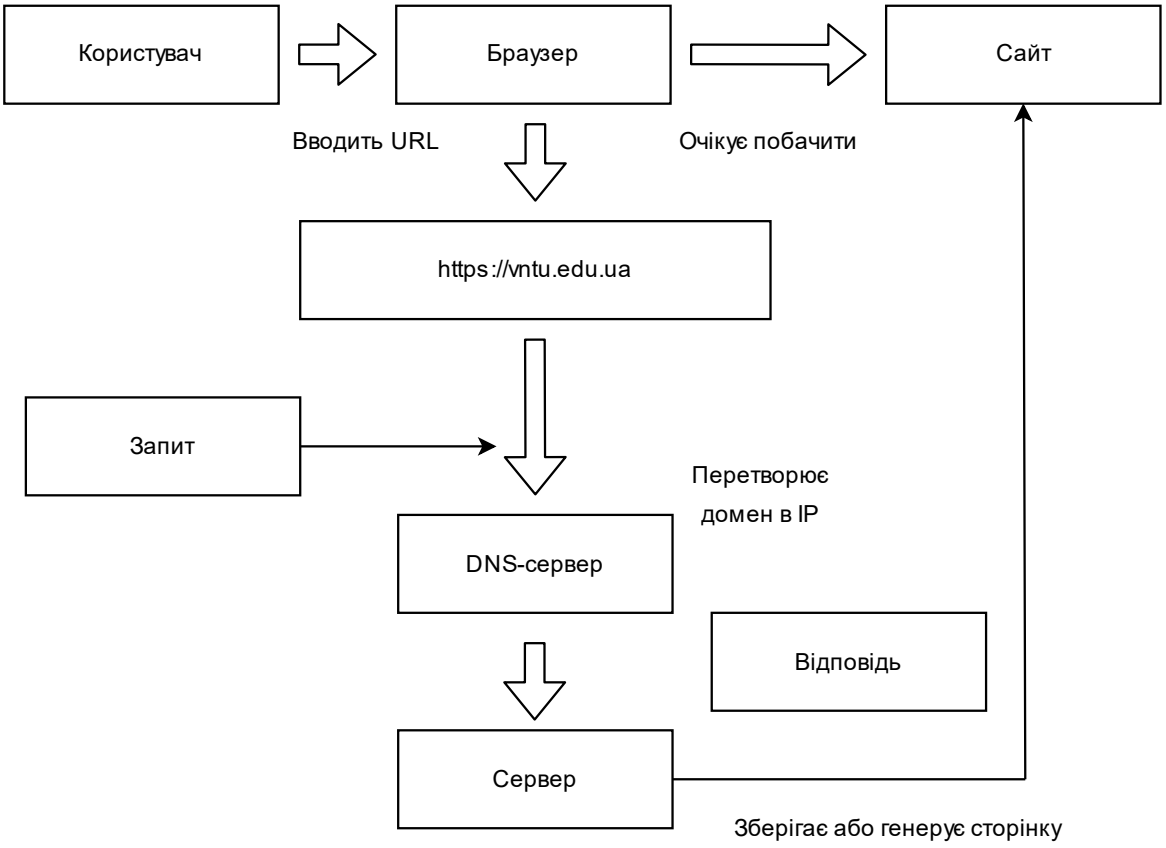


Рисунок 2.10 – Схема роботи веб-ресурсів

В цілому процес відображення сайту для користувача розбивається на 4 етапи:

– браузер зчитує адресу сайту (URL), який ввів користувач;
– браузер надсилає запит до серверу, на якому знаходиться веб-сайт;
– браузер здійснює обробку відповіді;
– браузер відображає відповідну сторінку користувачеві.

Далі буде розглянуто кожен етап окремо.

Очевидно, що код веб-сайту не зберігається на комп'ютері і повинен бути отриманим з серверу, де він зберігається. Наприклад, коли користувач хоче отримати доступ до сайту кафедри захисту інформації ВНТУ він повинен ввести в адресну стрічку відповідний домен – zi.vntu.edu.ua. Однак, сервери, що зберігають вихідні коди сайтів ідентифікуються не за доменним ім'ям, а за IP-адресами. Браузер відправляє запит до серверу з відповідною IP-адресою. Зазвичай, IP-адреси виглядають так (мова про IPv4): 194.146.142.4. Для того щоб перетворити доменне ім'я у IP-адресу використовуються відповідні сервери імен, або DNS сервери (Domain Name System). Їх основною задачею є перетворення доменних імен у IP-адреси. Для простоти можна уявляти DNS сервери як таблицю, де зберігаються значення Домен → IP-адреса. Таким чином, коли користувач хоче отримати доступ до сайту, що має доменне ім'я zi.vntu.edu.ua браузер перетворює його у відповідну IP-адресу, для даного випадку це 194.146.142.4.

Після того, як IP-адресу було визначено браузер відправляє запит до відповідного серверу. Запит може містити у собі різні параметри, які збираються воєдино (URL, метод, IP-адреса, параметри та ін). Приклад запиту до сайту кафедри захисту інформації ВНТУ наведено на рис. 2.11.

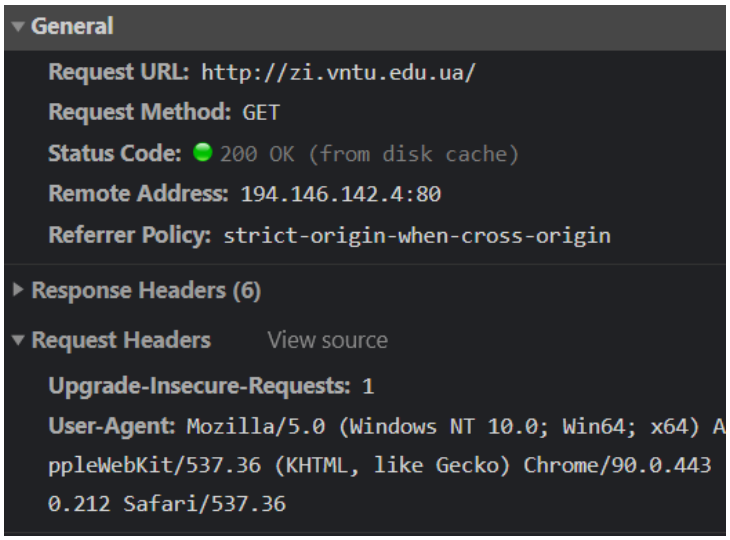


Рисунок 2.11 – Приклад запиту до сайту кафедри захисту інформації ВНТУ

Дані відправляються за допомогою спеціального протоколу – HTTP. HTTP (Hyper Text Transfer Protocol) – це стандартизований прокол, який визначає, як повинні виглядати запит та відповідь, які дані можуть в себе включати та як відбувається обробка запитів.

Після отримання запиту, сервер здійснює його обробку (шукає дані, які хоче побачити користувач) та надсилає відповідь. Приклад відповіді серверу кафедри захисту інформації ВНТУ наведено на рис. 2.12.

Рисунок 2.12 – Відповідь серверу кафедри захисту інформації ВНТУ

Так як і запит, тіло відповіді може містити велику кількість різноманітних даних. Так, наприклад, це може бути HTML-розмітка, яку потім браузеру потрібно вивести на екран. Відповідний приклад наведено на рис. 2.13.

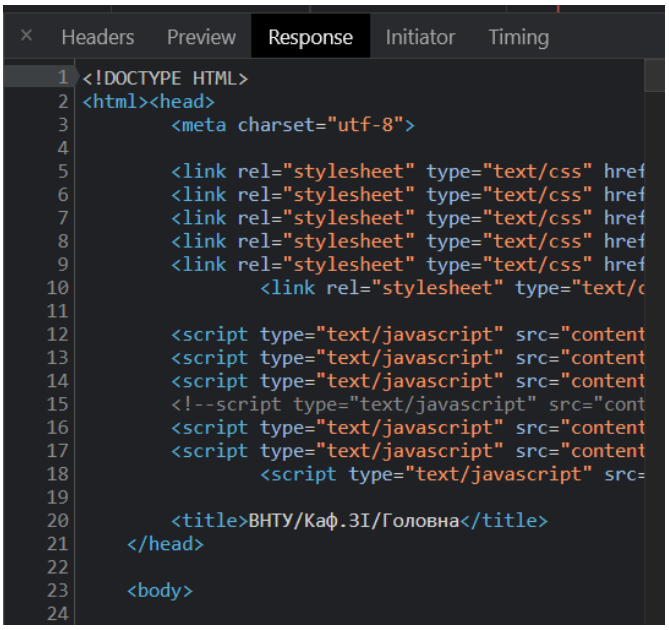


Рисунок 2.13 – Відповідь, що містить HTML-розмітку

Також, відповідь може містити відповідні CSS-файли для надання сторінці стилів, або JS-файли для надання інтерактивності. Вміст відповіді визначають розробники веб-ресурсу.

Наступним кроком є обробка отриманого запиту. Після отримання відповіді потрібно здійснити її обробку для подальшого виведення користувачеві. Браузер перевіряє тип даних, які були надіслані сервером і в залежності від цього вирішує, що потрібно робити далі. Так, на рисунку 2.11 вказано, що надіслано тип text/html. Це можна побачити у полі Content-Type. Таким чином, браузер знає, що даний документ потрібно відобразити користувачу і здійснює відображення коду з рисунку 2.13.

Після відображення основної структури веб-сторінки браузер перевіряє, чи не потрібно додати стилі (CSS) або логіку (JavaScript) на сторінку. Якщо вказано, що сторінка повинна містити щось додатково, то надсилаються повторні запити для отримання відповідних даних. Їх приклад наведено на рис. 2.14.

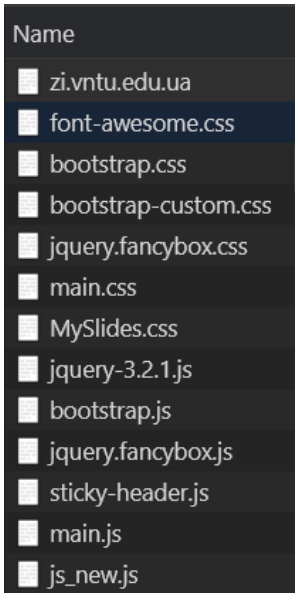


Рисунок 2.14 – Приклад додаткових файлів, які отримує браузер

Таким чином, браузер додає отримані файли до документу, що створює запланований вигляд для веб-сторінки.

Саме так користувач отримує доступ до потрібного ресурсу. Сайти, що будуть створені у 3 розділі працюватимуть за таким же принципом.

Кожна зі складових спроектована для виконання конкретної задачі. Попередньо необхідно навести перелік даних, які будуть використовуватися у системі, щоб мати чітке розуміння того, як зберігати інформацію, які засоби та методи повинні бути використані при збиранні інформації про мішень та мати готовий шаблон для генерації звітів. Далі наведено перелік усіх типів даних, які будуть використовуватися у системі:

– номер телефону;
– електронна адреса;
– паспортні дані;
– профілі у соціальних мережах;
– залученість особи до судових засідань;
– пошук паролів по логіну у соціальних мережах або електронній адресі;
– фото з соціальних мереж.

Далі буде наведено алгоритм збору даних по декільком категоріям. Спочатку розглянемо роботу з номером телефону цілі. Після введення номера телефону у системі, користувач надсилає запит до Backend-частини, яка починає обробку даного запиту. Так, перевіряється країна реєстрації номеру, мобільний оператор. Після отримання цієї інформації починається перевірка наявності даного номеру у базах телефонних номерів. Так, при отриманні інформації про власника номеру все зберігається у базу даних, після чого система повинна перейти до наступного етапу. Далі необхідно перевірити, чи є відомі зв’язки між номером мобільного телефону та соціальними мережами або електронною адресою. При наявності інформації необхідно зберегти її до бази даних. Алгоритм роботи системи при пошуку інформації по номеру мобільного телефона наведено на рис. 2.15.

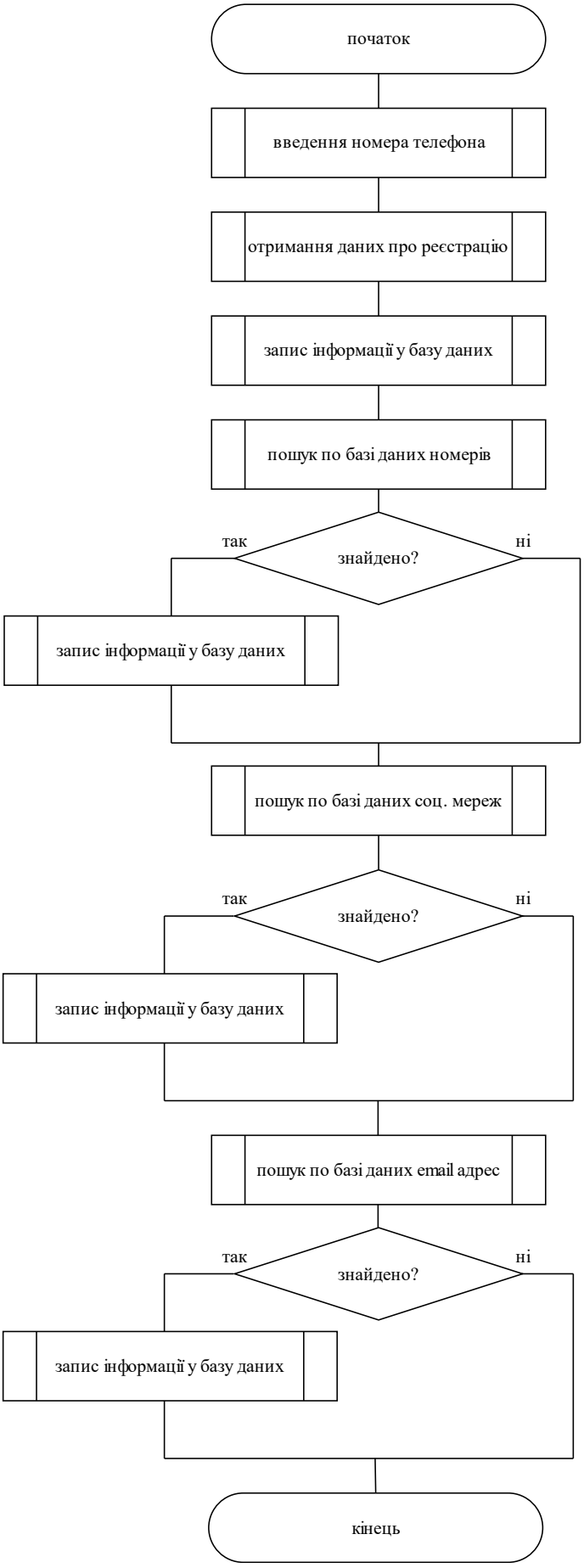


Рисунок 2.15 – Алгоритм збору інформації за номером телефону

Аналогічно працює і механізм збору даних за допомогою електронної адреси та профілем у соціальних мережах. Вся різниця між алгоритмами – у порядку виконання дій. Тепер розглянемо алгоритм перевірки на залученість особи до судових засідань.

Для цього доцільно використати Opendatabot [21]. Opendatabot– це українська компанія, що надає доступ до державних даних з основних публічних реєстрів для громадян та бізнесу. Так, наприклад, він дозволяє знайти інформацію про людей, які брали участь у судових засіданнях або знаходяться у розшуку. Приклад роботи даного сервісу наведено на рис. 2.16.

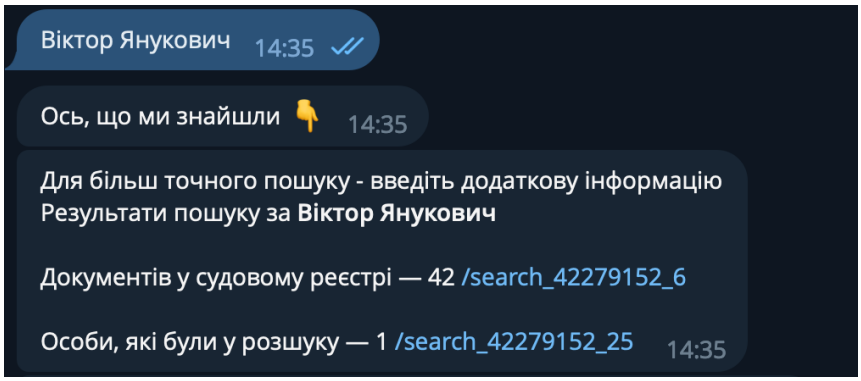


Рисунок 2.16 – Приклад роботи Opendatabot

Все, що потрібно для того, щоб отримати потрібну інформацію про людину – це мати її ім’я та прізвище. Після цього, необхідно зайти на веб-сайт компанії або в один з її чат-ботів (наявні у Telegram, Viber), чи скористатися API для розробників, передавши ресурсу дані людини, пошук інформації про яку ви хочете здійснити. Далі, додаток автоматично надасть дані з відкритих реєстрів. Таким чином, маючи ім’я та прізвище користувача – можна отримати цінну інформацію, яку потім можна використати при подальшому дослідженні. Алгоритм взаємодії системи з Opendatabot наведено на рис. 2.17.

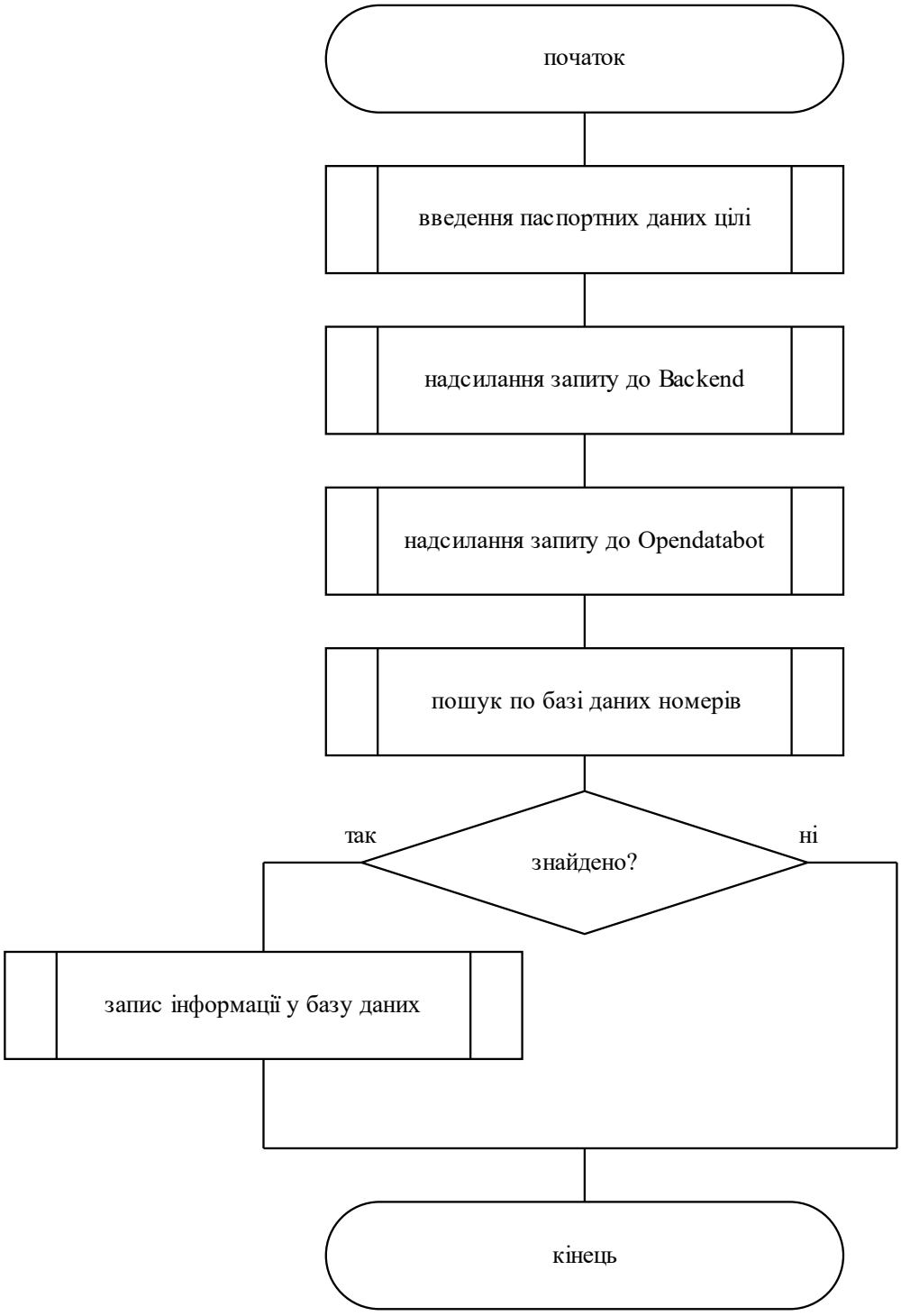


Рисунок 2.17 – Алгоритм взаємодії з Opendatabot

Таким чином, при наявності відповідної інформації про людину, вона буде додана до бази даних системи та з’явиться у звітах.

Також необхідно розуміти, що не всі дані є взаємопов’язаними, тому система повинна вміти виконувати пошук незалежно від наявності інших типів даних. Так, наприклад, не завжди існує кореляція між паспортними даними та

номером телефону, або соціальними мережами та залученістю особи до судових засідань. Саме тому, запустити пошук одразу по всім параметрам – неможливо. Передбачається, що користувач повинен мати певний набір даних з самого початку, згідно з якими буде здійснюватися подальший пошук.

2.4 Розробка Frontend-частини

Тепер, коли відомо, як працюють веб-сайти та наведено приклади майбутньої роботи, необхідно визначити, що саме повинна робити Frontend-частина.

Для реалізації поставленої задачі необхідно реалізувати веб-сайт, який повинен містити наступний функціонал:

- додавання нових цілей для пошуку;
- видалення вже наявних цілей;
- редагування наявних цілей;
- надсилання запитів на старт нового пошуку до Backend-частини;
- відображення інформації, зібраної з пошуку;
- редагування зібраної інформації;
- перегляд звітів, згенерованих Backend-частиною.

Тепер, коли у нас визначені основні функції, необхідно визначити, як Frontend повинен взаємодіяти з Backend.

У веб-розробці для цього використовують AJAX запити, які є непомітними для звичайного користувача, та дозволяють взаємодіяти з веб-сайтом без оновлення сторінки.

AJAX це абревіатура від Asynchronous JavaScript and XML. Дана група технологій дозволяє браузерам асинхронно відправляти та отримувати дані з веб-серверу. Завдяки даній технології сайти виглядають більш швидкими та подібні до додатків на персональному комп’ютері. Саме завдяки AJAX користувацькі інтерфейси виглядають живими, додатково покращується користувацький досвід, адже для завантаження нової порції даних не потрібно

перезавантажувати сторінку. Різницю між звичайним HTTP-запитом та AJAX-запитом наведено на рис. 2.18.

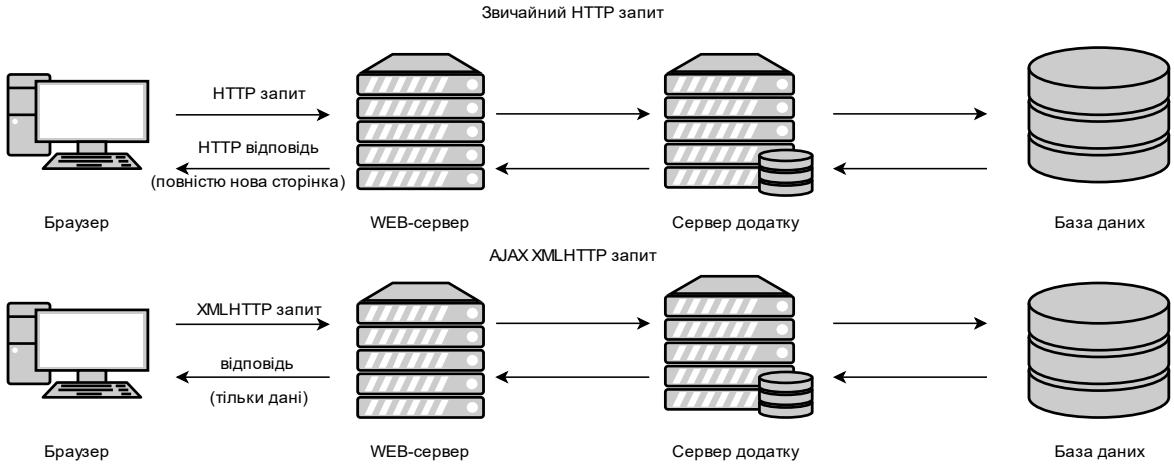


Рисунок 2.18 – Порівняння HTTP- та AJAX-запитів

Перша схема показує, як виглядає звичайний HTTP-запит, де сервер повертає повну сторінку (включаючи HTML, CSS та JavaScript). Після отримання сторінки, браузер «видаляє» поточну сторінку і завантажує нову, щоб продемонструвати її користувачеві. Звичайно, користувач не може виконувати будь-які дії на сторінці, а тільки чекати, поки завантажиться нова.

Друга схема демонструє роботу AJAX-запиту. Браузер використовує JavaScript, щоб відправити XMLHttpRequest (XHR) об’єкт на сервер. Даний об’єкт містить інформацію про те, яка інформація запитується. Таким чином, сервер відповідає тільки тими даними, які були запитані. Коли сервер надіслав відповідь з потрібними даними, браузер використовує JavaScript для отримання даних, їх обробки та оновити тільки частину сторінки, яка була змінена. Саме цей процес і забезпечує асинхронність на задньому плані без обов’язкового перезавантаження сторінки. Дана технологія використовується на більшості популярних ресурсів. Наприклад, коли користувачі додають коментарі до постів на Facebook. Після відправки сторінка не перезавантажується, а нові коментарі одразу видно на сторінці.

Тому, доцільно використати AJAX-запити для взаємодії Frontend та Backend -частин.

Тепер, коли визначено алгоритм роботи Frontend-частини інформаційної технології для збирання інформації про мішень доцільно перейти до опису роботи Backend-частини.

2.5 Розробка Backend-частини

З інформації поданої раніше, можна зробити висновки, що Frontend-частина це все, з чим взаємодіє користувач у браузері. Backend, в свою чергу, це все, що працює на сервері, тобто не у браузері або на комп'ютері користувача. Зазвичай, цей сервер під'єднаний до мережі Інтернет і відповідає за обробку, зберігання, редагування, відправлення інформації згідно із запитами від інших комп'ютерів.

Для розробки Backend-частини можна використовувати надзвичайно велику кількість інструментів, які дозволяють створити веб-сервер. Більшість з сучасних мов програмування дозволяють створювати веб-сервери з мінімальними зусиллями. Так, наприклад, можна обирати серед PHP, Ruby, Java, JavaScript (Node.JS). Завдяки гнучкості можна обирати не тільки мову програмування, а і будь-які системи керування базами даних, такі як MySQL, PostgreSQL, MongoDB, Cassandra, Redis та ін.

Як вже було визначено раніше, для створення даної Backend-частини використано Node.JS та PostgreSQL.

Незважаючи на те, що Node.JS дозволяє створювати веб-сервери без додаткових бібліотек, все ж доцільно скористатися фреймворком Express. Він дозволяє пришвидшити розробку за рахунок вже побудованих абстракцій над вбудованими функціями Node.JS, що значно зменшує кількість коду, який необхідно написати, для повноцінного функціонування додатку. Express – це мінімалістичний і гнучкий фреймворк для Node.JS додатків. Він забезпечує наступні механізми:

– легко створювати будь-які обробники HTTP-запитів;

– здійснювати обробку запитів на різних URL;
– встановлювати популярні значення для веб-додатків, такі як порт підключення та місцезнаходження шаблонів відображення;
– додатково обробляти запити за допомогою «middleware».

Не дивлячись на те, що сам по собі Express доволі мінімалістичний фреймворк, спільнота створила до нього багато додаткових пакетів, які дозволяють ще більше пришвидшити розробку. Існують сторонні бібліотеки для роботи з cookies, сесіями, авторизацією, платіжними системами та ін.

Таким чином, створений веб-сервер для інформаційної технології повинен мати наступні можливості:

– відображати список усіх наявних «цілей»;
– додавати нові «цілі»;
– редагувати вже наявні «цілі» у системі;
– видаляти вже наявні «цілі» у системі;
– виконувати пошук інформації про «мішень» згідно з встановленими параметрами;
– генерувати звіти про «мішень» у PDF форматі.

Далі необхідно визначити механізм роботи Backend-частини для даної інформаційної технології. Виходячи з того, як працює Frontend-частина, очевидно, що дана реалізація повинна містити обробник запитів, що надсилається за допомогою AJAX. Так, наприклад, розглянемо послідовність роботи інформаційної технології під час додавання нової «цілі» у систему:

– користувач вводить дані про мішень;
– Frontend-частина надсилає запит з наявною інформацією;
– Backend-частина отримує запит, починає його обробку;
– дані, які ввів користувач зберігаються у базу даних;
– Backend-частина відправляє повідомлення про те, що мішень було додано у систему;
– Backend-частина починає пошук згідно зі встановленими параметрами;

– Frontend сторінка з інформацією про мішень поступово оновлюється, по мірі надходження нових даних.

Відповідно, схема роботи наведена на рис. 2.19.

Рисунок 2.19 – Загальна схема роботи системи

Тепер, коли відома загальна схема роботи необхідно більш детально розглянути сутності, які наявні у системі та кінцеві точки, з якими буде взаємодіяти Frontend-частина.

Основна сутність системи – мішень, тому саме її опис буде наведено далі. Згідно з наведеною раніше інформацією про мішень необхідно збирати наступні дані:

- номер телефону;
- електронна адреса;
- паспортні дані;
- профілі у соціальних мережах;
- залученість особи до судових засідань;
- пошук паролів по логіну у соціальних мережах або електронній адресі;
- фото з соціальних мереж.

Виходячи з цього, необхідно створити сутність Target, яка містить наступні поля:

- phoneNumber: string;
- email: string;
- passportDetails: string;
- socialNetworks: array of objects;
- courtDecisions: array of objects;
- availablePasswords: array of objects;
- photos: array of strings.

Поле phoneNumber повинне містити номер телефону, має тип String (стрічка). После email повинне містити електронну адресу, має тип String. Поле passportDetails повинне містити інформацію про паспортні дані, має тип String. Поле socialNetworks повинне містити об’єкт з посиланнями на соціальні мережі, елементи об’єкту мають тип String. Поле courtDecisions повинне містити масив об’єктів, які стосуються участі цілі у судових засіданнях. Поле availablePasswords повинне містити потенційні паролі від соціальних мереж цілі. Поле photos повинне містити посилання на фото цілі з соціальних мереж, елементи масиву мають тип String. Детальний опис сутностей та зв’язок між ними наведено на UML-діаграмі (рис. 2.20).

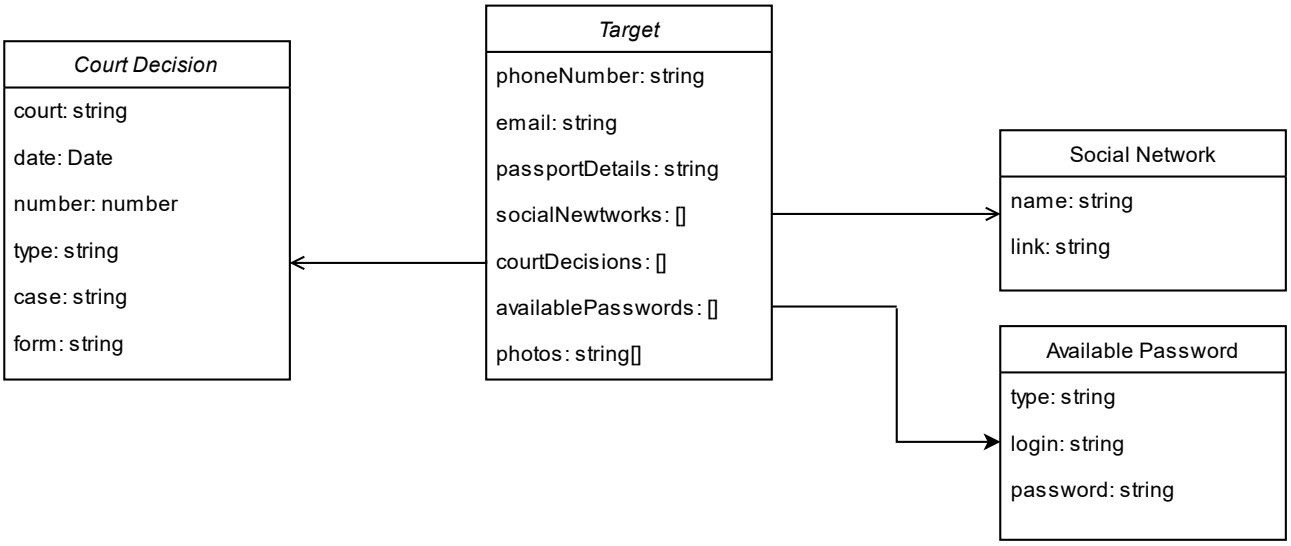


Рисунок 2.20 – Зв’язок між сутностями, що наявні у системі

Тепер, коли відома схема роботи Frontend-частини та Backend-частини необхідно визначити, як можна використовувати отриману інформацію про мішень.

Отже, у даному розділі виконано аналіз сценаріїв проведенння атак.

Запропоновано метод збирання інформації про мішень, що враховує етап проведення атаки. Наведено приклади використання отриманої інформації для створення фішингових повідомлень. Розроблено архітектуру програмного засобу, що використовується для збирання інформації про мішень, окремо описано роботу модулів для Frontend- та Backend-частини. Наведено схему роботи Backend-частини, наведено схему роботи Frontend-частини. Описано сутність Target, яка є основною сутністю у додатку. Наведено її зв’язок з іншими сутностями.

3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Обґрунтування вибору засобів програмної реалізації

Як вже було визначено раніше, для програмної реалізації інформаційної технології доцільно використати JavaScript для Frontend-частини та Node.JS з базою даних PostgreSQL для Backend-частини. JavaScript на даний момент є єдиною мовою програмування, яка дозволяє взаємодіяти з користувачем у браузері, тому наразі альтернатив для додання інтерактивності на Frontend-частині не існує. Саме тому збір потрібної інформації для створення цифрового відбитку, її обробка, створення геш-значення та відправка на сервер відбувається з використанням мови програмування JavaScript. HTML та CSS, як і JavaScript, є стандартами для WEB-додатків. Аналогічно, наразі не існує альтернатив, що дозволяють використовувати інші методи створення сторінок для глобальної мережі. Саме тому модуль для демонстрації програмного засобу реалізовано за допомогою HTML та CSS.

Node.JS є одним з багатьох альтернативних варіантів створення WEB-серверу [22]. Node.JS – це середовище виконання коду JavaScript поза межами браузера. Таким чином, за допомогою цього реалізується парадигма «JavaScript для всього». Вона передбачає використання однієї мови програмування для розробки WEB-додатків замість використання різних мов програмування для роботи з Backend та Frontend.

Основною перевагою використання JavaScript є можливість виконувати код паралельно. Це досягається за рахунок обробника подій. Наприклад, програма може просто чекати, поки користувач виконає якусь дію, а потім викликати необхідну функцію. Більш того, процеси асинхронні: додаток може запитати інформацію з бази даних на самому початку роботи. І поки база даних повертає 40 відповідні значення, програма на Node.JS далі виконує необхідні дії. Коли відповідь отримано – додаток виконає її обробку. При розробці масштабних продуктів на Node.JS кількість таких паралельних подій і обробників може сягати кількох сотень. У такі моменти платформа створює

нескінченний цикл, в якому по черзі надає процесорний час для кожної функції. Саме такий підхід і надає серйозну перевагу при створенні серверних додатків. Наприклад, раніше алгоритм роботи web-серверу виглядав таким чином:

- на сторінці необхідно відобразити аватар та нікнейм користувача;
- для цього сервер робить запит до бази даних, щоб отримати необхідну інформацію;
- поки база даних не надала відповідь – сервер чекає;
- коли відповідь з аватаром та нікнеймом отримано, сервер продовжує роботу, завантажуючи далі сторінку сайту;
- в кінці, наприклад, знову робиться запит до бази даних, щоб отримати фонове зображення;
- сервер очікує необхідне зображення;
- в цей час, сторінка не працює, тому що не завантажилась до кінця.

Таким чином, очевидно, що даний підхід має ряд суттєвих недоліків. Так, на кожен запит користувача потрібно виділяти ресурси, щоб підтримувати з’єднання з базою даних. Якщо таких запитів багато – ресурси поступово починають закінчуватись, що призводить до сповільнення роботи.

Для порівняння, алгоритм роботи серверів з використанням Node.JS:

- на сторінці необхідно відобразити аватар та нікнейм користувача;
- для цього сервер робить запит до бази даних, щоб отримати необхідну інформацію, а сам продовжує формувати сторінку далі;
- поки аватар не надіслано сервер виконав всі необхідні задачі і зрозумів, що потрібно отримати фонове зображення;
- сервер робить відповідний запит до бази даних, в той же час отримуючи попередню відповідь;
- аватар та нікнейм додаються на сторінку, після чого отримується і фонова картинка, що так само додається у потрібне місце.

Таким чином, всі матеріали розташовуються на потрібних місцях, після чого сторінку можна показувати користувачу.

Очевидно, що швидкість завантаження значно більша, адже збірка виконувалася паралельно. Крім створення web-додатків дану платформу використовують для створення будь-яких сервісів, де потрібен постійний обмін інформацією з користувачами:

– чати;
– соціальні мережі;
– системи сумісної роботи над проектами;
– онлайн-редактори текст.

Останнім часом набирає популярності тренд використання Node.JS для «інтернету речей» – розумних чайників, ваг та іншої побутової техніки. Тут Node.JS допомагає управляти цими приборами і створювати сервери, які можуть обробляти багато запитів одночасно. Саме через ряд суттєвих переваг Node.JS у якості основної платформи було обрано багатьма відомими компаніями:

– Walmart – велика мережа американських роздрібних магазинів;
– LinkedIn – соціальна мережа ділових контактів;
– Uber – таксі;
– E-bay – аукціон-платформа для продажу товарів по усьому світу;
– GitHub – репозиторій проектів для програмістів;
– Netflix – американський провайдер медійних послуг;
– Trello – сервіс для керування проектами.

PostgreSQL – це об’єктно-реляційна система керування базами даних (СКБД), найбільш розвинена серед відкритих СКБД у світі [23]. Має відкритий вихідний код і є альтернативою комерційним базам даних. Ранні версії системи були засновані на старій програмі PostgresUniversity, створеній університетом Берклі: так з’явилась назва PostgreSQL. СКБД дозволяє гнучко керувати базами даних (БД). За допомогою них можна створювати, модифіковувати або видаляти записи, відправляти транзакції – набір з декількох послідовних запитів на особливій мові запитів SQL.

Далі наведено перелік основних користувачів PostgreSQL:

- Backend-розробники, яким потрібно взаємодіяти з базами даних при роботі з серверною частиною сайту;
- адміністратори та розробники баз даних – спеціалісти, які займаються обслуговуванням та підтримкою роботи бази даних;
- DevOps-інженери, які обслуговують інфраструктуру проекту.

Таким чином, СКБД PostgreSQL є актуальною для тих, хто розробляє серверну частину сайтів, або займаються обслуговуванням та модифікацією вже існуючої БД. Далі наведено перелік основних цілей, для яких можна використовувати PostgreSQL:

- гнучкий доступ до баз даних, їх організація та зберігання;
- керування записами у базах даних: створення, редагування і видалення, оновлення версій;
- перегляд потрібної інформації з бази по запиту, наприклад для того, щоб відправити її на сайт або інтерфейс додатку для відображення;
- відправка транзакцій, послідовних запитів, зібраних у своєрідний скрипт;
- налаштування і контроль доступу до тої чи іншої інформації, групування користувачів згідно з їхніми правами доступу;
- контроль версій і організація одночасного доступу до бази із різних місць так, щоб уникнути збоїв;
- захист інформації від можливих витоків і втрат;
- контроль стану бази в цілому.

Прийнято вважати, що PostgreSQL є безкоштовним аналогом Oracle Database. Обидві системи адаптовані під великі проекти з великим навантаженням. Однак, є різниця – вони по-різному зберігають дані.

Традиційно популярні СКБД – реляційні. Це означає, що дані, які в них зберігаються, представляються у вигляді записів, що пов’язані один з одним відношеннями (relations). Пов’язані списки можуть мати між собою ті чи інші відношення, так і формується таблиця. Існує ще одна популярна модель – об’єктна. У даному випадку дані представляються у вигляді об’єктів, їх

атрибутів, методів та класів. PostgreSQL – це об’єктно-реляційна СКБД. Це означає, що вона підтримує як об’єктний, так і реляційний підходи.

Ще одна особливість даної СКБД – це підтримка великої кількості типів записів інформації. Це не тільки стандартні цілочислені значення, числа з плаваючою комою, текст і бульові значення, а й грошові, геометричні, лічильники, бінарні та інші типи. При необхідності до неї можна підключити підтримку типів даних, які потрібні в кожному конкретному проекті.

В PostgreSQL можна писати свої власні функції – користувацькі блоки коду, які виконують ті чи інші дії. Ця можливість є у багатьох СУБД, однак в PostgreSQL підтримується більше мов, чим аналоги. Крім стандартних SQL, в PostgreSQL можна писати і на C і на C++, Java, Python, PHP, Lua і Ruby. Вона підтримує V8 – один із рушіїв JavaScript, тому його також можна використовувати разом з PostgreSQL.

Важлива особливість PostgreSQL – можливість отримати доступ до бази з декількох пристроїв. В СКБД реалізована клієнт-серверна архітектурна, коли база даних зберігається на сервері, а доступ до неї здійснюється з клієнтських комп’ютерів. Так, наприклад, реалізовуються різноманітні сайти. Одна з можливих складностей – ситуація, коли декілька людей одночасно модифікують базу і потрібно уникнути можливих конфліктів. В PostgreSQL для цього використовується технологія MVCC – Multiversion Concurrency Control, багаторівневе керування паралельним доступом. Кожен користувач отримує снапшот – «знімок» бази, до якої вносяться зміни. Тільки після фіксації транзакції вони вносяться до основної бази даних. Поки людина вносить зміни, вони не помітні для інших користувачів. Таким чином, конфлікти не виникають, як і немає необхідності блокувати читання та запис.

PostgreSQL – це програмне забезпечення з відкритим вихідним котом, яке розповсюджується по вільній ліцензії. Це означає, що будь-який розробник може переглянути, як написана система, або запропонувати для неї свої правки. СКБД розробляється спільнотою ентузіастів, і в певному сенсі нікому не належить, а значить, що її можна без обмежень використовувати у своїх проектах.

Незважаючи на те, що це СКБД з відкритим вихідним кодом, проект відомий великою кількістю тестів. Кожна версія системи з’являється в доступі тільки після повної перевірки, тому СКБД є дуже стабільною. Згідно з незалежними автоматизованими дослідженнями, у вихідному коді PostgreSQL є одна помилка на 39 000 стрічок коду. Це в п’ять разів менше, ніж в MySQL.

Як вже було сказано раніше, для розробки Frontend-частини доцільно використати бібліотеку React.

Бібліотека React була випущена компанією Facebook у 2013 році [24]. Через 3 роки після релізу, у 2016, сайт для розробників StackOverflow провів опитування, де 50 000 тисяч працівників розказали про свою роботу та професійні уподобання. Вже тоді було зрозуміло, що дана технологія є надзвичайно перспективною. Результати опитування наведено на рис. 3.1.

За допомогою React розробники створюють веб-додатки, які змінюють відображення без перевантаження сторінки. Завдяки цьому додатки швидко реагують на дії користувачів, наприклад, заповнення форм, додавання фільтрів, додавання товарів в корзину і так далі. React використовують для відображення компонентів користувацього інтерфейсу. Також бібліотека може повністю керувати Frontend-частиною. У випадку з React використовують з іншими бібліотеками для керування станом і роутингом, наприклад, Redux і ReactRouter. Одна з ключових особливостей React – універсальність. Дану бібліотеку можна використовувати як для розробки веб-сайтів у браузері, так і для мобільних платформ за допомогою ReactNative. Даний принцип називається LearnOnce, WriteAnywhere (Навчись один раз, пишіть де завгодно).Ще одна важлива особливість бібліотеки – декларативність. За допомогою React розробники описують, як компоненти інтерфейсу виглядають у різних станах. Декларативний підхід зменшує кількість коду і робить його більш зрозумілим. Для побудови інтерфейсів React використовує компоненти, це ще одна ключова особливість бібліотеки. Кожен компонент повертає частину користувацього інтерфейсу зі своїм станом. Поєднуючи компоненти, програміст створює

завершений інтерфейс веб-додатку. Далі буде описано основні переваги та недоліки використання React.

Віртуальна об’єктна модель документу. Об’єктна модель документа (DOM) визначає деревовидну структуру HTML-документа, який надсилається клієнту сервером після відповідного запиту. DOM представляє веб-сторінку у об’єктивно-орієнтованому форматі, щоб мови програмування могли з нею взаємодіяти. Браузер регулярно перевіряє будь-які зміни в DOM, оновлюючи її відповідним образом. Зміни об’єктної моделі документом провокується множиною факторів, наприклад введенням даних користувачем, HTTP-запитом, отриманням даних від API та ін. Браузер оновлює DOM кожен раз, коли сторінка змінюється. Оскільки DOM сучасних сайтів надзвичайно великі, оновлення займає багато часу, що суттєво зменшує загальну продуктивність веб-додатку. Замість повільної і незручної взаємодії безпосередньо з реальною об’єктною моделлю документа, React взаємодіє з її полегшеною копією – віртуальний DOM, тому реальний DOM оновлюється тільки після взаємодії з віртуальним DOM. Виходячи з цього, віртуальний DOM забезпечує розробників двома серйозними перевагами.

Перша перевага – ефективність. Оновлення всього DOM, щоб зробити веб-сторінку «реактивною» – надзвичайно неефективно, оскільки використовує занадто багато ресурсів. Тому, при зміні веб-сторінку (наприклад, внаслідок запиту або дії користувача) React оновлює спеціальний віртуальний DOM. React зберігає в пам’яті дві версії віртуального DOM – оновлений віртуальний DOM і його резервну копію, створену до оновлення. Після оновлення React порівнює обидві версії між собою, щоб найти елементи, які змінилися, а потім – оновлює виключно ту частину реального DOM, яка змінилась. На перший погляд, даний процес здається занадто складним і затратним, однак, це займає набагато менше часу, ніж оновлення реальної об’єктної моделі документа повністю. Таким чином робота з DOM оптимізується.

Друга перевага – висока продуктивність. Одна з найважливіших цілей для будь-якого стартапу – написати веб-додаток швидким та продуктивним,

забезпечивши найкраще обслуговування для потенційних клієнтів. Віртуальний DOM, на відміну від реального DOM, займає мало місця та швидко оновлюється, тим самим підвищуючи продуктивність додатку. Віртуальний DOM дозволяє сторінці миттєво отримувати відповіді від сервера і відображати оновлення. Наприклад, Facebook використовує технологію віртуального DOM для оновлення чатів і стрічки новин користувача без перезавантаження сторінки.

При роботі з ReactJS створюються багаторазові компоненти: частіше за все, компонент користувацького інтерфейсу можна використовувати в других частинах коду або в різних проектах, практично без змін. Більш того, розробникам React-додатків доступні бібліотеки готових компонентів з відкритим вихідним кодом. Завдяки їм, час розробки суттєво скорочується, що надзвичайно важливо для стартапів, яким необхідно економити не тільки гроші, а й час.

Компанія Facebook приклала багато зусиль, щоб зробити React потужним інструментом для покращення користувацького інтерфейсу будь-який веб-додатків. Крім того, корпорація активно продовжує покращувати бібліотеку, роблячи її більш приємною та ефективною технологією. Однак, не тільки Facebook створювали React: випустивши бібліотеку з відкритим вихідним кодом у 2013, Facebook активно заохочував розробників покращувати бібліотеку. Нині React – результат спільної роботи розробників з усього світу.

Однак, при використанні React розробники можуть зіштовхнутися і з певними недоліками. Один із основних – складності пошукової оптимізації та інтеграції з пошуковими системами. Пошукова оптимізація (SEO) надзвичайно важлива, адже веб-додатку необхідно отримувати трафік і залучати нових клієнтів. Для кращого розуміння складності пошукової оптимізації на React, необхідно розуміти, як Google індексує веб-сторінки.

Веб-сторінки індексуються спеціальними ботами Google. Ці боти сканують вміст веб-сторінок і зберігають інформацію в індексі Google. Коли користувач запитує певну інформацію, то Google перевіряє дані, які зберігаються в індексі, щоб надати найбільш релевантну інформацію. Боти Google можуть легко

індексувати HTML-сторінки. Однак з JavaScript сторінками це не так легко, адже динамічні веб-додатки, в тому числі створені на React, проходять ускладнену процедуру індексації, на відміну від статичних веб-сторінок. Таким чином, сторінка React-додатки може бути проіндексована неправильно або індексування може зайняти дуже багато часу. Обидва випадки змушують ботів Google покинути сторінку сайту.

ReactJS – це JavaScript-бібліотека, що містить у собі набір інструментів для створення інтерфейсу веб-додатків. Саме це не дозволяє називати React повноцінним фреймворком, хоча таке представлення доволі розповсюджене. React – це не інструмент «все в одному» для створення цілого додатку. Виходячи з цього, якщо слідувати шаблону проектування MVC (Model-View-Controller), то React буде відповідати тільки за представлення. Дві інші частини – модель і контролер, створюються за допомогою додаткових інструментів. Якщо ви використовуєте React в проекті, то вам необхідно інтегрувати додаткові інструменти для маршрутизації, написання інтерфейсів прикладного програмування (API) та інших частин повноцінного веб-додатку.

Незважаючи на недоліки, React протягом багатьох років доводить свою ефективність для створення інтерфейсів, тому і застосовується так широко. Тепер, коли було обґрунтувано вибір технологій можна переходити безпосередньо до реалізації інформаційної технології.

3.2 Розробка програмного засобу

Відповідно до визначеного раніше алгоритму роботи програмного засобу було створено:

- Frontend-частину;
- Backend-частину.

Оскільки дана розробка має у своєму складі дві частини, доцільно буде скомпонувати їх в одну папку, розділивши на складові компоненти (рис 3.1).

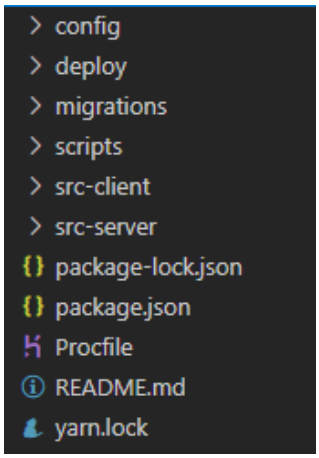


Рисунок 3.1 – Складові частини розробки

Так, до папки src-client вносяться всі файли, що необхідні для побудови Frontend-частини, вміст даної папки наведено на рис. 3.2.

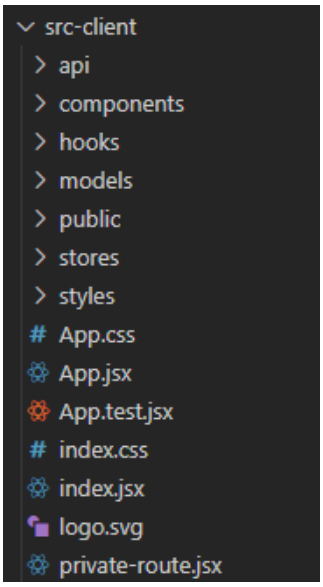


Рисунок 3.2 – Вміст папки src-client

Оскільки, для розробки використовується React – усі JavaScript файли мають розширення JSX. Усі файли, що відповідають за відображення інформації знаходяться у папці components, у папці api знаходиться файл, який дозволяє полегшити роботу з Backend-частиною, у папці models описано вигляд сутності Target (рис. 3.3).

Рисунок 3.3 – Вигляд сутності Target

Даний код реалізує сутність, яка була описана у другому розділі (див. рис. 2.20).

У папці public знаходяться статичні файли, які відображатиме браузер. У папці stores знаходяться функції, що дозволяють полегшити взаємодію з API, надавши додаткові абстракції. У папці styles знаходяться стилі, які відповідають за розміщення та стилізування елементів на екрані. Тепер, коли було описано Frontend-частину можна перейти до опису Backend-частини.

Вміст папки, до якої входять усі файли, що відповідають за Backend-частину наведено на рис. 3.4.

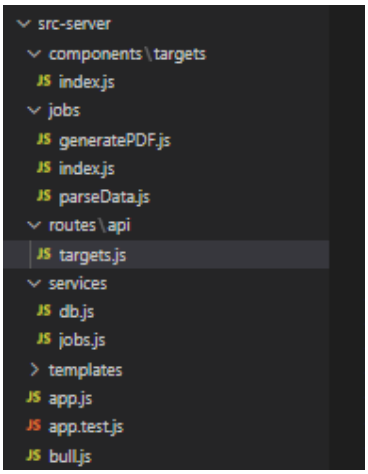


Рисунок 3.4 – Структура папки src-server

У папці components міститься файл, що відповідає за взаємодію з базою даних для Target. У папці jobs знаходяться усі Jobs, які потенційно може виконувати система. GeneratePDF генерує PDF документ, зберігає його та відправляє користувачу для відображення. У папці routes знаходиться опис ендпоінтів, які наявні у системі. У папці servicesмістяться файли, які відповідають за підключення до бази даних та опис методів, що застосовуються до кожної Job. Файл app.js описує загальну структуру додатку, налаштовує підключення до бази даних, формат відображення даних, вмикає необхідні ендпоінти та обробляє помилки, що виникають під час роботи додатку.

Перший метод дозволяє створювати нову мішень. Він отримує електронну адресу, номер телефону або паспортні дані, після чого ця інформація записується до бази даних, а потім передається у Job, яка виконує парсинг інформації. Також у цей час відправляється відповідь, що мішень було додано.

Другий метод дозволяє отримати всі цілі, які наявні у системі. Він викликає компонент target та його метод get, який знаходить у базі даних всі цілі.

Далі наявні методи, які дозволяють переглянути інформацію про мішень, відредагувати інформацію (електронну адресу, номер телефону, паспортні дані) або видалити мішень.

Передостанній метод дозволяє окремо запустити парсинг для цілі, у разі, якщо парсинг після додавання був невдалий, або є необхідність перевірити інформацію ще раз.

Останній метод дозволяє згенерувати PDF звіт, щоб користувач міг завантажити його собі на комп’ютер та використовувати у своїх цілях.

Також, доцільно продемонструвати файл, який безпосередньо відповідає за взаємодію з базою даних (рис. 3.7).

index.js

src-server > components > targets > index.js > ...

```
module.exports = (app) => {
  const db = app.get("db");
  const { targets } = db;
  const module = {};

  // Create

  module.create = async (row) => {
    if (!row) throw new Error("No row data given");
    delete row.id;
    return targets.save(row);
  };

  // Get all

  module.get = async () => targets.find();

  // Get One
  module.getOneById = async (id) => users.findOne({ id });
  module.getOneByEmail = async (email) => users.findOne({ email });
  module.getOneByPhone = async (phone) => users.findOne({ phone });

  // Update

  module.update = async (id, row) => {
    if (!Number(id)) throw new Error("No id given");
    row.id = id;
    return targets.save(row);
  };

  // Delete

  module.delete = async (id) => {
    if (!Number(id)) throw new Error("No id given");
    return targets.destroy({ id });
  };

  return module;
};
```

Рисунок 3.7 – Вигляд файлу, що відповідає за взаємодію з базою даних

У базі даних знаходяться сутності Target, їх опис було наведено у другому розділі (див. рис. 2.20).

Так, саме ці функції використовує роутер, коли надсилає відповідь у кожному ендпоінті. Завдяки використанню ORM немає необхідності писати чисті SQL-запити, що дозволяє уникати проблем, таких як SQL-ін’єкції.

Перший метод відповідає за створення нової цілі. Він приймає дані (row) та записує їх до бази даних, повертаючи новий об’єкт.

Другий метод відповідає за повернення усіх цілей. Він використовує функцію find без параметрів, що дозволяє знайти усі наявні у системі варіанти цілей.

Далі є перелік методів, які дозволяють знайти одну мішень по якомусь конкретному параметру, це може бути ID в базі даних, адреса електронної пошти або номер телефону.

Передостанній метод призначений для оновлення даних про мішень у базі даних. Він оновлює інформацію по ID, повертаючи вже оновлений об’єкт.

Останній метод дозволяє видалити мішень з бази даних.

Після того як було продемонстровано реалізацію доцільно перейти до тестування програмного засобу.

3.3 Тестування програмного засобу

Для тестування працездатності додатку необхідно обрати мішень та здійснити пошук всієї інформації про неї. Оскільки зібрана інформація буде демонструватися – система буде перевірена на авторі розробки. Для цього необхідно перейти на сайт, щоб додати нову ціль (рис. 3.8).

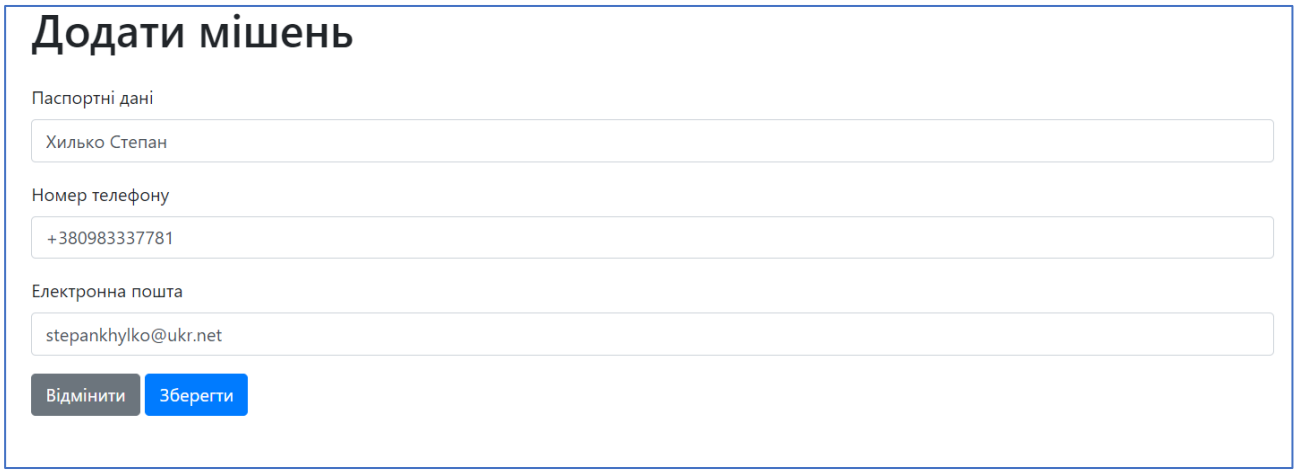


Рисунок 3.8 – Додавання цілі у систему

У прикладі заповнені всі поля, однак, необхідно заповнювати хоча б одне. Після цього необхідно натиснути кнопку «Зберегти». Система поверне користувача на головний екран. Вигляд головного екрану наведено на рис. 3.9.

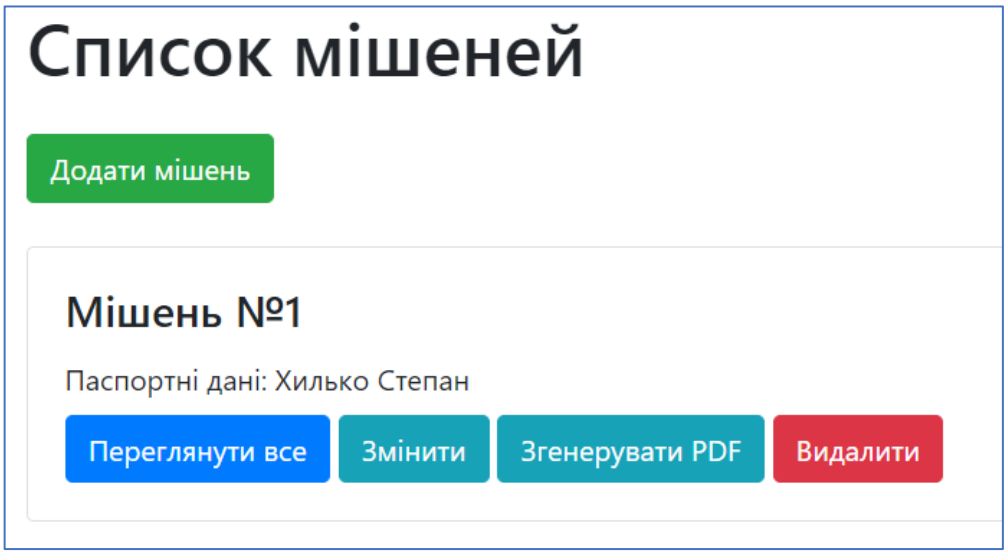


Рисунок 3.9 – Вигляд головного вікна розробки

На головній сторінці наведено перелік усіх наявних у системі цілей. Так, їх можна редагувати, змінювати, згенерувати PDF-файл або видалити. Системі необхідний певний час, щоб система зібрала усю інформацію про ціль. Результат роботи наведено на рис 3.10.

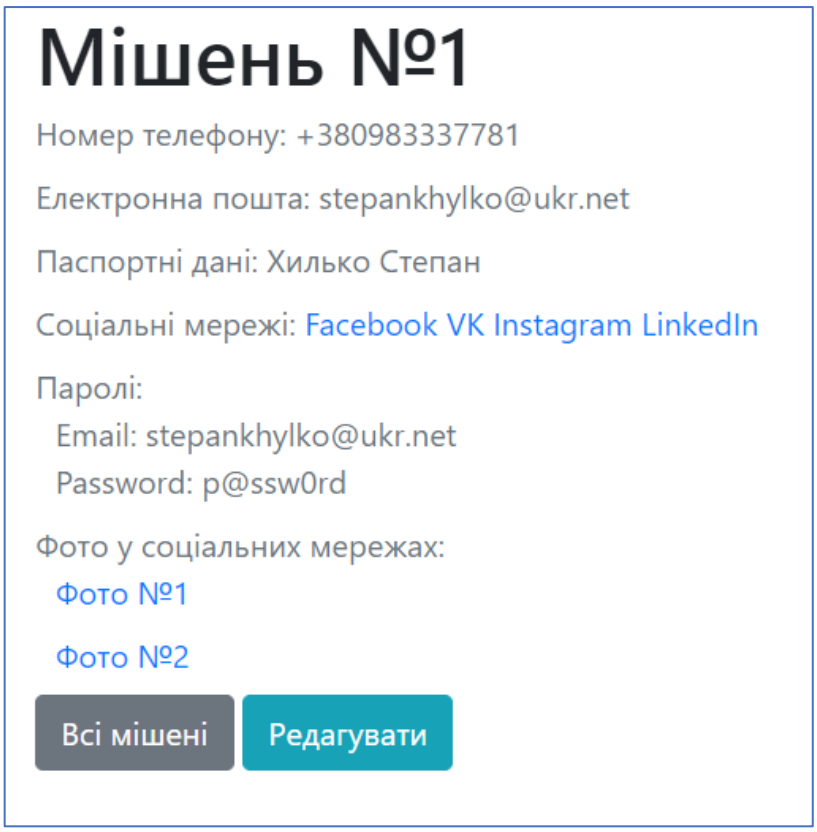


Рисунок 3.10 – Результат пошуку інформації про ціль

Таким чином, про автора розробки було знайдено наступну інформацію:

– посилання на соціальні мережі;
– номер телефону;
– скомпроментовані паролі;
– фото з соціальних мереж;

Дана інформація насправді відповідає дійсності.

Так, за попередньої згоди, було проведено аналіз одного з одногрупників. Результати пошуку наведено на рис. 3.11.

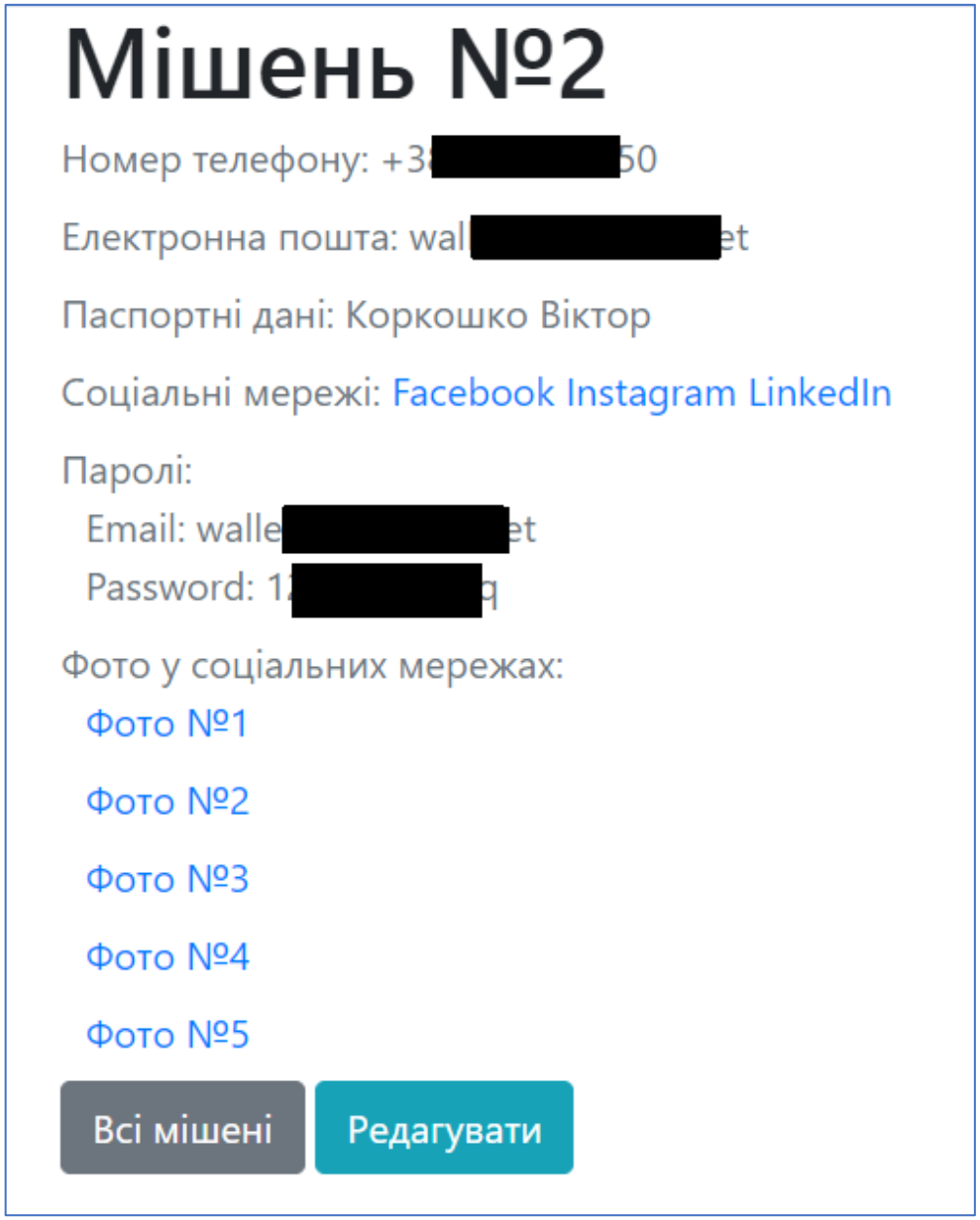


Рисунок 3.11 – Результати пошуку інформації про ціль

Також, за попередньої згоди, було проведено аналіз ще однієї особи. Результати наведено на рис. 3.12.

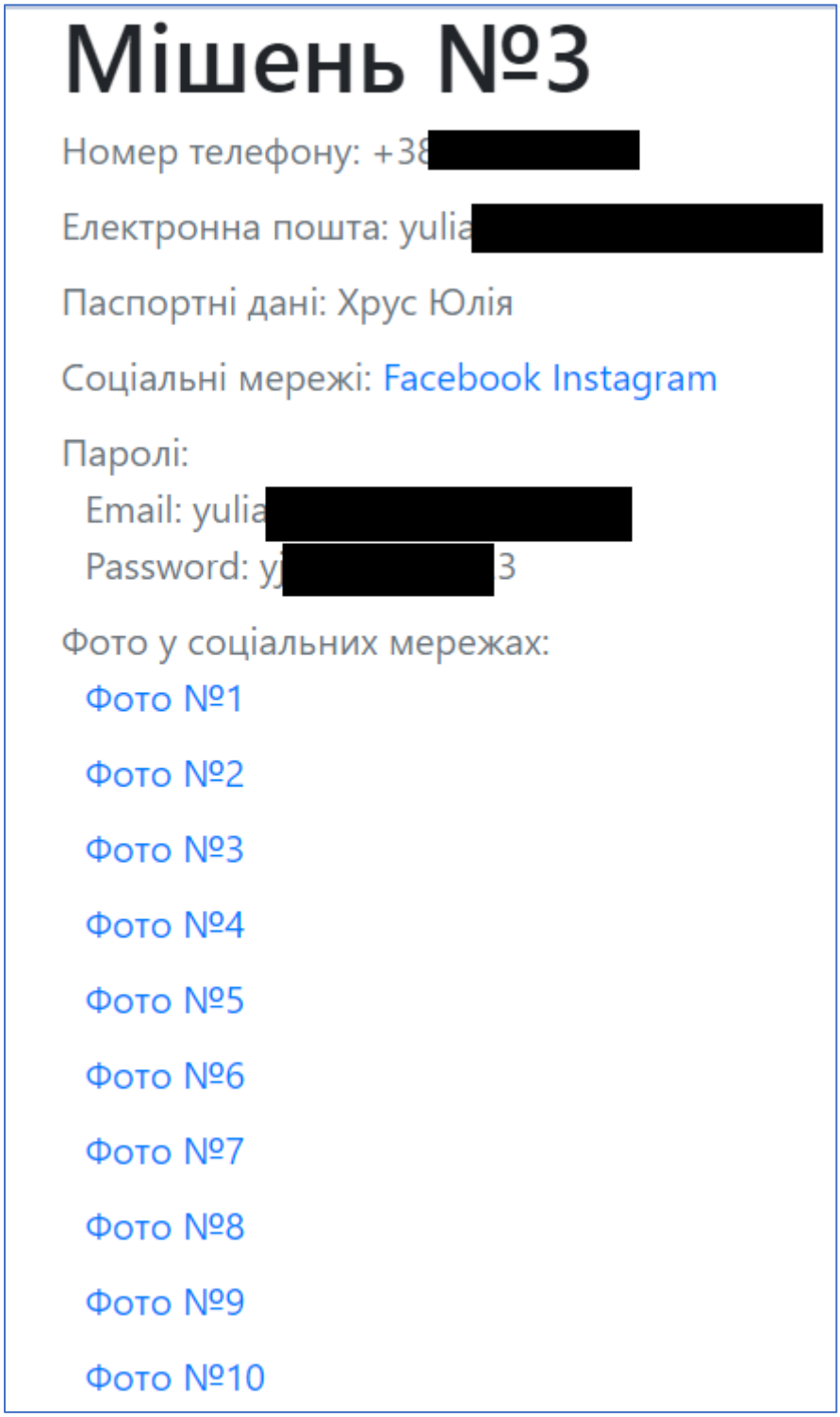


Рисунок 3.12 – Результати пошуку інформації про ціль

Так, у даному розділі було перевірено роботу усіх основних функцій засобу для збирання інформації про мішень. Додавання нової цілі, відправка даних на сервер, збір інформації про ціль, редагування та видалення цілей у системі, генерування PDF-звітів. Додаток працює правильно, безвідмовно.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота «Метод та засіб збирання інформації про мішень під час інформаційної війни» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [25].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів реалізацію продукту.	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 4.2.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	2	2	3
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	2	2	2
7. Ринкові перспективи (конкуренція)	4	4	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	41	42	40
Середньоарифметична сума балів $СБ_c$	41,0		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [25].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб збирання інформації про мішень під час інформаційної

війни» становить 41,0бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості (Вн) для нового технічного рішення розрахуємо за формулою [26]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \quad , \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

αі – коефіцієнт, який враховує питому вагу і-го технічного показника в загальній якості розробки. Коефіцієнт αі визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

βі – відносне значення і-го технічного показника якості нової розробки.

Відносні значення βі для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{нi}}{I_{ai}} \quad , \tag{4.2}$$

де Іні та Іна – чисельні значення конкретного і-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \quad (4.3)$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
швидкість	бал	9	9,8	1,09	0,35
точність	%	90	97	1,07	0,25
зручність	%	90	97	1,07	0,15
надійсність	%	83	94	1,13	0,1
маштабованість	%	70	85	1,21	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}09 \cdot 0{,}35 + 1{,}07 \cdot 0{,}25 + 1{,}07 \cdot 0{,}15 + 1{,}13 \cdot 0{,}1 + 1{,}21 \cdot 0{,}15 = 1{,}10.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,10 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Метод та засіб збирання інформації про мішень під час інформаційної війни», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп,

науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [25]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 16300{,}00 \cdot 44 / 22 = 32600{,}00 \ грн.$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	16300,00	740,91	44	32600,00
Інженер-розробник програмного забезпечення	16150,00	734,09	40	29363,64
Технік	6825,00	310,23	20	6204,55
Всього				68168,18

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Метод та засіб збирання інформації про мішень під час інформаційної війни» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i– коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б)[25];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}65 / (22 \cdot 8) = 69{,}09$ *грн.*
$З_{р1} = 69{,}09 \cdot 6{,}00 = 414{,}56$ *грн.*

Таблиця 4.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Підготовка обладнання для розробки програмного забезпечення	6,00	2	1,10	69,09	414,56
Підготовка робочого місця розробника програмного забезпечення	4,23	3	1,35	84,80	358,69

Інсталяція програмного забезпечення	5,66	5	1,70	106,78	604,38
Налагодження програмних блоків	3,21	4	1,50	94,22	302,44
Всього					1680,08

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (68168,18 + 1680,08) · 10 / 100% = 6984,83 грн.

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (68168,18 + 1680,08 + 6984,83) · 22 / 100% = 16903,28 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни».

Витрати на матеріали ($М$), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де H_j– норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 … 1,15*);

B_j – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$M_1 = 3,0 \cdot 226,00 \cdot 1,1 - 0 \cdot 0 = 745,80$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний Line OFFICE білий	226,00	3,0	0	0	745,80
Папір офісний 80 г/м 10 кольорів по 10 аркушів Rainbow	80,00	3,0	0	0	264,00
Набір настільний 410 А 15 предметів	253,00	3,0	0	0	834,90
Набір настільний SCHOLZ	370,00	2,0	0	0	814,00
Тонер CANON LBP-3010/3020 (TSH87B) black	515,00	1,0	0	0	566,50
Диск оптичний CD-RW	21,00	3,0	0	0	69,30
USB флеш накопичувач Transcend 16Gb JetFlash 700 (TS64GJF700)	120,00	1,0	0	0	132,00
Всього					3426,50

4.3.3 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($K_в$)*, які використовують при проведенні НДР на тему «Метод та засіб збирання інформації про мішень під час інформаційної війни», розраховуємо, згідно з їхньою номенклатурою, за формулою.

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \quad (4.10)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$).

$К_в = 1 \cdot 270,00 \cdot 1,11 = 299,70$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Концентратор Speedlink SNAPPY SLIM USB Hub, 4-Port, USB 2.0, Passive, Black	1	270,00	299,70
Кабель для передачі даних SATA III 1.0m Cablexpert (CC-SATAM-DATA-XL)	1	48,00	53,28
Всього			352,98

4.3.4 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою.

$$В_{спец} = \sum_{i=1}^{k} Ц_i \cdot С_{пр.i} \cdot К_i \ , \quad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$С_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *2999,00 · 1 · 1,1 = 3298,90 грн.*

Отримані результати зведемо до таблиці.

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Маршрутизатор MikrotikhAPac (RBD52G-5HacD2HnD-TC)	1	2999,00	3298,90
Комутатор мережевий TP-Link TL-SG1005D	1	779,00	856,90
Всього			4155,80

4.3.5 Програмнезабезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою.

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг}$ = *105,00 · 1 · 1,11 = 116,55 грн.*

Отримані результати зведемо до таблиці.

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки: VisualStudioCode	1	105,00	116,55
Всього			116,55

4.3.6 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою.

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (32599{,}00 \cdot 2) / (2 \cdot 12) = 2716{,}58$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Asus X556U Процесор: IntelCore I5-6198DU Оперативна пам'ять: 16 Гб	32599,00	2	2	2716,58

Робоче місце інженера-розробника ПЗ	8752,00	5	2	291,73
Пристрої передачі даних	6810,00	2	2	567,50
Пристрій виводу інформації	6791,00	5	2	226,37
Оргтехніка	8300,00	4	2	345,83
Приміщення лабораторії	764000,00	20	2	6366,67
ОС Windows 10	8360,00	2	2	696,67
Прикладний пакет Microsoft Office 2016	7900,00	2	2	658,33
Всього				11869,68

4.3.7 Паливо та енергія для науково-виробничихцілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i– тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 6,20 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni}<1$;

η_i– коефіцієнт корисної дії обладнання, $\eta_i<1$.

$B_e = 0,04 \cdot 320,0 \cdot 6,20 \cdot 0,95 / 0,97 = 79,36$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Asus X556U Процесор: IntelCore I5-6198DU Оперативна пам'ять: 16 Гб	0,04	320,0	79,36

Робоче місце інженера-розробника ПЗ	0,15	320,0	297,60
Пристрої передачі даних	0,01	320,0	19,84
Пристрій виводу інформації	0,50	20,0	62,00
Оргтехніка	0,62	3,0	11,53
Інше	0,10	46,0	28,52
Всього			498,85

4.3.8 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Метод та засіб збирання інформації про мішень під час інформаційної війни» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою.

$$B_{св} = (З_о + З_р) \cdot \frac{H_{св}}{100\%}, \qquad (4.15)$$

де $H_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$B_{св}$ = (68168,18 + 1680,08) · 20 / 100% = 13969,65 грн.

4.3.9 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою.

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (4.16)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (68168,18 + 1680,08) · 30 / 100% = 20954,48 грн.

4.3.10 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \qquad (4.17)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (68168,18 + 1680,08) · 50 / 100% = 34924,13 грн.

4.3.11 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (4.17)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$В_{нзв} = (68168,18 + 1680,08) \cdot 120 / 100\% = 83817,91$ *грн.*

Витрати на проведення науково-дослідної роботи на тему «Метод та засіб збирання інформації про мішень під час інформаційної війни» розраховуємо як суму всіх попередніх статей витрат за формулою

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.18)$$

$В_{заг}$= *68168,18 +1680,08 +6984,83 +16903,28 +3426,50 +352,98 +4155,80 + 116,55 + 11869,68 +498,85 +13969,65 +20954,48 +34924,13 +83817,91 = 267822,90 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.19)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 267822,90 / 0,95 = 281918,84 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	2750	4350	7500	6000

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 15000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 12450,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 794,80 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [25]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.20)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ=30%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році ϑ=18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(794,80·15000,00+13244,80·2750)·0,83·0,3·(1-0,18/100%)=9871122,94 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(794,80·15000,00+13244,80·7100)·0,83·0,3·(1-0,18/100%)=21634929,13 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(794,80·15000,00+13244,80·14600)·0,83·0,3·(1-0,18/100%)=41917353,61 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(794,80·15000,00+13244,80·20600)·0,83·0,3·(1-0,18/100%)=58143293,20 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.21)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,25;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =9871122,94/(1+0,25)[1]+21634929,13/(1+0,25)[2]+41917353,61/(1+0,25)[3]+ +58143293,20/(1+0,25)[4]=7896898,35+13846354,65+21461685,05+23815492,89=67 020430,94 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.22}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 281918,84 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 281918,84 = 563837,68 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.23}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 67020430,94 грн;

PV – теперішня вартість початкових інвестицій, 563837,68 грн.

$E_{абс} = ПП - PV$ = 67020430,94 - 563837,68 = 66456593,26 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.24)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 66456593,26 грн;

PV – теперішня вартість початкових інвестицій, 563837,68 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+66456593,26/563837,68)^{1/4} = 2,30.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.25)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,12;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,38.

$\tau_{мін}$ = 0,12+0,38 = 0,5 < 2,30 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.26)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 2,30 = 0,43 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни» становить 41,0бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 1,10 рази.

Також термін окупності становить 0,43 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Метод та засіб збирання інформації про мішень під час інформаційної війни».

ВИСНОВКИ

З початком повномасштабного вторгнення, актуальним є проведення атак, зокрема і у кіберпросторі. Основна мета проведення таких операцій – вплив на настрої груп у суспільстві через конкретних її представників. Для успішного проведення такої кібератаки необхідно чітке формування інформаційного повідомлення, який би змусив конкретного користувача (мішень) виконати необхідні дії, наприклад, відкрити файл у додатку до листа, або перейти на фішингове посилання, або повірити у фейкову новину.

Розглянуто поняття інформаційної війни та її основні ознаки. Наведено приклади впливу відкритих даних на інформаційно-психологічні операції під час інформаційної війни. Розглянуто поняття розвідки з відкритих джерел, сфери використання її результатів. Проведено аналіз аналогічних засобів, у ході якого виділено їхні переваги та недоліки. З’ясовано, що існуючі програми часто не включають в себе всю необхідну інформацію, або не надають її у зручному для користування вигляді, що сповільнює та утруднює розгортання кібератаки. Поставлено завдання на розробку методу та засобу збору інформації про мішень, який би враховував етапи проведення інформаційної операції та надавав саме ту інформацію, яка потрібна на цьому етапі.

Проаналізувавши різні типи інформації, яку можна знайти про користувача, було виділено первинну інформацію. До неї входять: паспортні дані, номер телефону, електронна пошта. Так, з первинної інформації можна отримати вторинну, а саме: профілі у соціальних мережах, залученість особи до судових засідань, пошук паролів по логіну у соціальних мережах або електронній пошті, фото з соціальних мереж.

Запропоновано метод збирання інформації про мішень, що враховує етап проведення атаки. Наведено приклади використання отриманої інформації для створення фішингових повідомлень. Розроблено архітектуру програмного засобу, що використовується для збирання інформації про мішень, окремо описано роботу модулів для Frontend- та Backend-частини. Наведено схему роботи Backend-частини, наведено схему роботи Frontend-частини. Описано

сутність Target, яка є основною сутністю у додатку. Наведено її зв’язок з іншими сутностями.

Під час опису програмних технологій було доведено доцільність використання обраних програмних засобів та технологій, після чого розроблено структуру та архітектуру веб-додатку. Розроблено алгоритм роботи програми. Створено програмну реалізацію.

Проведено тестування, в результаті чого визначено відповідність поставленому завданню.

Оцінювання економічної доцільності системи для методу та засобу збирання інформації про мішень під час інформаційної війни показало, що дана розробка є доцільною, оскільки період окупності складає менше трьох років, що демонструє економічну перспективність отриманих результатів. Розроблену систему можна використовувати в організаціях, де є потреба до забезпечення підвищеного рівня безпеки.

Реалізований засіб можна використовувати при проведенні інформаційних операцій у кіберпросторі під час інформаційної війни для збирання інформації про певну особу (мішень). В подальшому планується введення нових джерел даних, підвищення швидкості та розширення інформації, яку здатна знайти система.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хилько С., Войтович. М. Збирання інформації про мішень під час інформаційної війни: Електронний ресурс. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/view/16911

2. Інформаційні війни в історії та сучасності: характерні ознаки новітніх протистоянь. URL: http://enpuir.npu.edu.ua/bitstream/handle/123456789/25591/Zhadko%2064-95.pdf?sequence=1 (дата звернення 10.09.2022).

3. Defense Intelligence Agency Expected to Lead Military's Use of Open Source Data. URL: https://www.wsj.com/articles/defense-intelligence-agency-expected-to-lead-militarys-use-of-open-source-data-11639142686 (дата звернення 15.09.2022).

4. Крим Archives – InformNapalm. URL: https://informnapalm.org/ua/category/news/crimea/ (дата звернення 30.09.2022).

5. The Battle of Ilovaisk. URL: https://ilovaisk.forensic-architecture.org/ (дата звернення 05.10.2022).

6. Як працює OSINT-розвідка? Від бізнесу до оборони України. URL: https://www.issp.training/post/yak-pratsyuye-osint-rozvidka-vid-biznes-analizu-do-oborony-ukrayiny (дата звернення 16.09.2022).

7. Що таке OSINT і як він допоміг викрити вбивства у Бучі. URL: https://explainer.ua/shho-take-osint-i-yak-vin-dopomig-vikriti-vbivstva-u-buchi/ (дата звернення 10.10.2022).

8. Як OSINT впливає на війну в Україні? URL: https://blog.iteducenter.ua/articles/osint/(дата звернення 15.10.2022).

9. Російські ІПСО – як Кремль психологічно тисне на українців? URL: https://24tv.ua/rosiyski-ipso-yak-kreml-psihologichno-tisne-ukrayintsiv-svit_n2210078 (дата звернення 16.10.2022).

10. Інформаційна війна приваблива тим, що невеликими ресурсами можна досягти великих політичних цілей. URL: https://hromadske.radio/podcasts/myslennia-bazova-funktsiia/informatsiyna-viyna-

pryvablyva-tym-shcho-nevelykymy-resursamy-mozhna-dosiahty-velykykh-politychnykh-tsiley-balaban (дата звернення 20.10.2022).

11. Що таке ІПСО, чому важливо це знати і які операції зараз проводить Росія проти України. URL: https://tyzhden.ua/shcho-take-ipso-chomu-vazhlyvo-tse-znaty-i-iaki-operatsii-zaraz-provodyt-rosiia-proty-ukrainy/ (дата звернення 23.10.2022).

12. Інформаційна війна – це не тільки фейки. URL: https://ms.detector.media/propaganda-ta-vplivi/post/29264/2022-03-31-informatsiyna-viyna-tse-ne-tilky-feyky/ (дата звернення 25.10.2022).

13. Інформаційна складова війни: як Росія намагається послабити підтримку Заходу. URL: https://www.radiosvoboda.org/a/informatsiyna-viyna-rosiyskyy-vplyv/31811302.html (дата звернення 26.10.2022).

14. Соціальна інженерія – як один із проявів кіберзлочинності. URL: http://buk-visnyk.cv.ua/news/1581/ (дата звернення 27.10.2022).

15. Соціальна інженерія: як шахраї використовують людську психологію в інтернеті. URL: https://www.radiosvoboda.org/a/socialna-inzhenerija-shaxrajstvo/29460139.html (дата звернення 30.10.2022).

16. Не вір нікому. Що таке соціальна інженерія та як не стати її жертвою. URL: https://kyivstar.ua/uk/cybersecurity/ne-vir-nikomu-shcho-take-socialna-inzheneriya-ta-yak-ne-staty-yiyi-zhertvoyu (дата звернення 01.11.2022).

17. Як захиститися від атак з елементами соцінженерії. URL: https://www.issp.ua/post/social-engineering (дата звернення 25.09.2022).

18. Google Hacking Test. URL: https://www.tutorialspoint.com/google_hacking_tests.htm (дата звернення 10.09.2022).

19. Фішинг – що це таке і яка мета фішингу? URL: https://www.eset.com/ua/support/information/entsiklopediya-ugroz/fishing/ (дата звернення 02.11.2022).

20. OSINT Dojo. URL: https://www.osintdojo.com/diagrams/linkedin (дата звернення 30.10.2022).

21. Опендатабот. URL: opendatabot.ua (дата звернення 10.11.2022).

22. JavaScript Підручник. Уроки для початківців. URL: https://w3schoolsua.github.io/js/index.html#gsc.tab=0 (дата звернення 11.11.2022).

23. What is PostgreSQL? Introduction, Advantages & Disadvantages. URL: https://www.guru99.com/introduction-postgresql.html (дата звернення 12.11.2022).

24. ReactJS. A brief history. URL: https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f (дата звернення 20.11.2022).

25. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

26. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

96

Додаток А

Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Метод та засіб збирання інформації про мішень під час інформаційної війни

Автор роботи: Хилько Степан Вікторович

Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 93,7%. Схожість – 6,3%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

☐ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Хилько С.В
(підпис) (прізвище, ініціали)

Керівник роботи _____ Войтович О.І
(підпис) (прізвище, ініціали)

Додаток Б

Текст програми

Index.js

```
module.exports = (app) => {
constdb = app.get("db");
const{ targets } = db;
constmodule = {};

  // Create

module.create = async (row) => {
if(!row) thrownewError("No rowdatagiven");
delete row.id;
returntargets.save(row);
  };

  // Get all

module.get = async () =>targets.find();

  // Get One
module.getOneById = async (id) =>users.findOne({ id });
module.getOneByEmail = async (email) =>users.findOne({ email });
module.getOneByPhone = async (phone) =>users.findOne({ phone });

  // Update

module.update = async (id, row) => {
if(!Number(id)) thrownewError("No idgiven");
    row.id = id;
returntargets.save(row);
  };

  // Delete

module.delete = async (id) => {
if(!Number(id)) thrownewError("No idgiven");
returntargets.destroy({ id });
  };

returnmodule;
};
```

Targets.js

```
constRouter = require("express-promise-router");
constTargets = require("../../components/targets");
constjobs = require("../../services/jobs");

module.exports = (app) => {
  constrouter = Router();
  consttargets = Targets(app);

  // Create Target

  router.post("/", async (req, res) => {
    constpayload = _.pick(req.body, "email", "phone", "passportDetails");

    constdata = awaittargets.create(payload);
    awaitjobs.perform("parseData", data);
    res.json(data);
  });

  // Get All Targets

  router.get("/", async (req, res) => {
    consttargets = awaittargets.get();

    res.json(targets);
```

```
  });

  // Get Single Target

  router.get("/:id", async (req, res) => {
    constdata = awaittargets.getOneById(req.params.id);
    res.json(data);
  });

  // Update One Target

  router.put("/:id", async (req, res) => {
    constpayload = _.pick(req.body, "email", "phone", "passportDetails");
    constdata = awaittargets.update(req.params.id, payload);
    res.json(data);
  });

  // Delete Target

  router.delete("/:id", async (req, res) => {
    constdata = awaittargets.delete(req.params.id);
    res.json(data);
  });

  // Start Target Parsing

  router.post("/startParsing/:id", async (req, res) => {
    constdata = awaittargets.getOneById(req.params.id);
    awaitjobs.perform("parseData", data);
    res.json({ status: 200, message: "Parsingstarted" });
  });

  // Generate PDF Report

  router.post("/generatePDF/:id", async (req, res) => {
    constdata = awaittargets.getOneById(req.params.id);
    constjobResult = awaitjobs.perform("generatePDF", data);
    res.json({ status: 200, message: "PDF Generated", jobResult });
  });
};
```

Db.js

```
constmassive = require('massive');
const{ db } = require('../../config/constants');

module.exports = async () => {
  // connecttoMassiveandgetthedbinstance
  constinstance = awaitmassive(db.connectionString, {
    // explicitlyspecifytheusedschemas
    allowedSchemas: [db.schema]
  });

  return{ db: instance };
};
```

Jobs.js

```
constQueue = require('bull');
const{ redisUrl } = require('../../config/constants');

constqueue = newQueue('job-queue', redisUrl);

asyncfunctiondiscard(pattern) {
  returnqueue.removeJobs(pattern);
}

asyncfunctionperform(name, data, opts) {
  if (opts&&opts.jobId) awaitdiscard(opts.jobId.toString());
  returnqueue.add(name, data, opts);
}

module.exports = {
  perform,
  discard,
};
```

App.js

```
constcreateError = require("http-errors");
constexpress = require("express");
constpath = require("path");
constlogger = require("morgan");
const DB = require("./services/db");
constapiTargets = require("./routes/api/targets");

module.exports = async () => {
  constapp = express();

  // viewenginesetup
  app.set("views", path.join(__dirname, "views"));
  app.set("viewengine", "ejs");

  app.use(logger("dev"));
  app.use(express.json());
  app.use(express.urlencoded({ extended: false }));

  const{ db } = await DB();
  app.set("db", db);

  app.use(express.static(path.join(__dirname, "../public")));

  // Enable CORS
  app.use((req, res, next) => {
    res.header("Access-Control-Allow-Origin", "*");
    res.header(
      "Access-Control-Allow-Headers",
      "Origin, X-Requested-With, Content-Type, Accept, Authorization"
    );
    res.header(
      "Access-Control-Allow-Methods",
      "PUT, POST, GET, DELETE, OPTIONS"
    );
    next();
  });

  app.get("/api/health", (req, res) =>res.json({ status: "ok" }));
  app.use("/api", apiTargets(app));

  // catch 404 andforwardtoerrorhandler
  app.use((req, res, next) => {
    next(createError(404));
  });

  // errorhandler
  // eslint-disable-next-lineno-unused-vars
  app.use((err, req, res, next) => {
    // sendtheerrorresponse
    console.error(err);
    res.status(err.status || 500);
    res.json(err);
  });

  returnapp;
};
```

Bull.js

```
constQueue = require('bull');
constjobs = require('./jobs');
const{ redisUrl } = require('../config/constants/redis');

constqueue = newQueue('job-queue', redisUrl);

queue.on('failed', (job, err) => {
  console.error(err);
});

Object.keys(jobs).forEach((name) => {
  queue.process(name, jobs[name]);
});
```

Package.json

```
{
  "name": "osint",
  "version": "0.7.0",
  "private": true,
  "scripts": {
    "start": "yarnserver:prod",
    "start:aws": "yarnbuild&&yarndb:migrate&&yarnserver:prod",
    "build": "nodescripts/build.js &&yarnlint",
    "bull": "nodesrc-server/bull.js",
    "deploy": "node ./deploy/deploy-node.js",
    "lint": "eslint 'src-client/**/*.{js,jsx}' 'src-server/**/*.js'",
    "seed": "yarndb:seed",
    "client:dev": "nodescripts/start-client-dev.js",
    "server:dev": "nodemon --watchsrc-server --ignore 'src-server/jobs/*.json' scripts/start-server-
dev.js",
    "server:prod": "nodescripts/start-server-prod.js",
    "dev": "npm-run-all -p server:devclient:dev",
    "pg-migrate": "node-pg-migrate",
    "db:migrate": "nodescripts/db-migrate",
    "db:seed": "yarndb:migrate&&nodescripts/db-seed.js",
    "test:client": "nodescripts/test.js --env=jsdom",
    "test:server": "nodescripts/test.js --config=config/jest/server.json",
    "update-version": "yarnversion --no-git-tag-version --patch"
  },
  "dependencies": {
    "@aws-sdk/client-s3": "^3.19.0",
    "@aws-sdk/s3-request-presigner": "^3.19.0",
    "@babel/core": "7.14.6",
    "@fortawesome/fontawesome-svg-core": "^1.2.36",
    "@fortawesome/free-solid-svg-icons": "^5.15.4",
    "@fortawesome/react-fontawesome": "^0.1.16",
    "@pmmmwh/react-refresh-webpack-plugin": "0.4.2",
    "@svgr/webpack": "5.5.0",
    "axios": "^0.21.0",
    "babel-eslint": "^10.1.0",
    "babel-jest": "^27.0.2",
    "babel-loader": "^8.2.2",
    "babel-plugin-named-asset-import": "^0.3.7",
    "babel-preset-airbnb": "^5.0.0",
    "bfj": "^7.0.2",
    "bootstrap": "^5.0.1",
    "browserslist": "^4.16.1",
    "bull": "^4.0.0",
    "camelcase": "^6.2.0",
    "case-sensitive-paths-webpack-plugin": "2.3.0",
    "chart.js": "^3.7.0",
    "cheerio": "^1.0.0-rc.10",
    "css-loader": "4.3.0",
    "dotenv": "^10.0.0",
    "dotenv-expand": "^5.1.0",
    "ejs": "^3.1.6",
    "eslint": "^7.28.0",
    "eslint-config-airbnb": "^18.2.1",
    "eslint-config-react-app": "^6.0.0",
    "eslint-plugin-flowtype": "^5.7.2",
    "eslint-plugin-import": "^2.23.4",
    "eslint-plugin-jest": "^24.3.6",
    "eslint-plugin-jsx-a11y": "^6.4.1",
    "eslint-plugin-react": "^7.24.0",
    "eslint-plugin-react-hooks": "^4.2.0",
    "eslint-plugin-testing-library": "^4.6.0",
    "eslint-webpack-plugin": "^2.5.4",
    "express": "^4.17.1",
    "express-promise-router": "^4.1.0",
    "extract-text-webpack-plugin": "^3.0.2",
    "faker": "^5.5.3",
    "file-loader": "6.1.1",
    "fs-extra": "^9.0.1",
    "html-webpack-plugin": "4.5.0",
    "http-errors": "^1.8.0",
    "identity-obj-proxy": "^3.0.0",
    "jest": "^27.0.4",
    "jest-circus": "^27.0.4",
    "jest-resolve": "^27.0.4",
    "jest-watch-typeahead": "^0.6.4",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.20",
```

```
    "massive": "^6.9.0",
    "mini-css-extract-plugin": "0.11.3",
    "mobx": "^6.3.2",
    "mobx-react-lite": "^3.2.0",
    "mobx-state-tree": "^5.0.1",
    "moment": "^2.29.1",
    "morgan": "^1.10.0",
    "multer": "^1.4.4",
    "node-pg-migrate": "^5.9.0",
    "nodemailer": "^6.7.0",
    "nodemon": "^2.0.6",
    "npm-run-all": "^4.1.5",
    "object-assign": "4.1.1",
    "optimize-css-assets-webpack-plugin": "5.0.4",
    "passport": "^0.4.1",
    "passport-jwt": "^4.0.0",
    "passport-local": "^1.0.0",
    "postcss-flexbugs-fixes": "4.2.1",
    "postcss-loader": "3.0.0",
    "postcss-normalize": "8.0.1",
    "postcss-preset-env": "6.7.0",
    "postcss-safe-parser": "5.0.2",
    "promise": "8.0.3",
    "prompts": "2.4.0",
    "prop-types": "^15.7.2",
    "puppeteer": "^10.4.0",
    "qs": "^6.10.1",
    "react": "^17.0.1",
    "react-app-polyfill": "^2.0.0",
    "react-chartjs-2": "^4.0.0",
    "react-dev-utils": "^11.0.1",
    "react-dom": "^17.0.1",
    "react-icons": "^4.3.1",
    "react-refresh": "^0.9.0",
    "react-router-dom": "^5.2.0",
    "resolve": "1.11.0",
    "resolve-url-loader": "^3.1.2",
    "sass-loader": "8.0.2",
    "semver": "7.3.2",
    "style-loader": "1.3.0",
    "supertest": "^6.1.3",
    "terser-webpack-plugin": "4.2.3",
    "url-loader": "4.1.1",
    "webpack": "4.44.2",
    "webpack-dev-server": "3.11.0",
    "webpack-manifest-plugin": "2.2.0",
    "workbox-webpack-plugin": "5.1.4",
    "xlsx": "^0.17.4"
  },
  "devDependencies": {
    "@aws-sdk/client-elastic-beanstalk": "^3.19.0",
    "archiver": "^5.3.0",
    "envalid": "^6.0.2",
    "enzyme": "^3.11.0",
    "enzyme-adapter-react-16": "^1.15.6",
    "enzyme-to-json": "^3.6.2",
    "husky": "^4.3.0"
  },
  "husky": {
    "hooks": {
      "pre-commit": "yarnlint"
    }
  },
  "jest": {
    "roots": [
      "<rootDir>/src-client"
    ],
    "collectCoverageFrom": [
      "src/**/*.{js,jsx,ts,tsx}",
      "!src/**/*.d.ts"
    ],
    "setupFiles": [
      "react-app-polyfill/jsdom"
    ],
    "testMatch": [
      "<rootDir>/src-client/**/__tests__/**/*.{js,jsx,ts,tsx}",
      "<rootDir>/src-client/**/*.{spec,test}.{js,jsx,ts,tsx}"
    ],
    "testEnvironment": "jsdom",
    "transform": {
```

```
    "^.+\\.(js|jsx|mjs|cjs|ts|tsx)$": "<rootDir>/node_modules/babel-jest",
    "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js",
    "^(?!.*\\.(js|jsx|mjs|cjs|ts|tsx|css|json)$)": "<rootDir>/config/jest/fileTransform.js"
  },
  "transformIgnorePatterns": [
    "[/\\\\]node_modules[/\\\\].+\\.(js|jsx|mjs|cjs|ts|tsx)$",
    "^.+\\.module\\.(css|sass|scss)$"
  ],
  "modulePaths": [],
  "moduleNameMapper": {
    "^react-native$": "react-native-web",
    "^.+\\.module\\.(css|sass|scss)$": "identity-obj-proxy"
  },
  "moduleFileExtensions": [
    "web.js",
    "js",
    "web.ts",
    "ts",
    "web.tsx",
    "tsx",
    "json",
    "web.jsx",
    "jsx",
    "node"
  ],
  "watchPlugins": [
    "jest-watch-typeahead/filename",
    "jest-watch-typeahead/testname"
  ],
  "resetMocks": true
},
"eslintConfig": {
  "extends": "airbnb",
  "env": {
    "browser": true,
    "node": true,
    "jest": true
  },
  "rules": {
    "camelcase": 0,
    "import/prefer-default-export": 0,
    "jsx-a11y/alt-text": 0,
    "max-len": 0,
    "no-console": 0,
    "radix": 0,
    "react/button-has-type": 0,
    "react/jsx-props-no-spreading": 0,
    "react/destructuring-assignment": 0,
    "react/prefer-stateless-function": 0,
    "react/prop-types": 0,
    "react/state-in-constructor": 0,
    "no-underscore-dangle": 0,
    "jsx-a11y/click-events-have-key-events": 0,
    "jsx-a11y/no-static-element-interactions": 0,
    "react/jsx-filename-extension": 0,
    "jsx-a11y/anchor-is-valid": 0,
    "react/forbid-prop-types": 0,
    "jsx-a11y/no-noninteractive-element-interactions": 0,
    "no-nested-ternary": 0,
    "class-methods-use-this": 0,
    "jsx-a11y/control-has-associated-label": 0,
    "jsx-a11y/label-has-associated-control": 0,
    "no-alert": 0,
    "no-param-reassign": 0,
    "global-require": 0,
    "comma-dangle": 0
  }
},
"proxy": "http://localhost:5000",
"babel": {
  "presets": [
    "airbnb"
  ]
},
"browserslist": {
  "production": [
    ">0.2%",
    "notdead",
    "notop_miniall"
  ],
```

```
    "development": [
      "last 1 chromeversion",
      "last 1 firefoxversion",
      "last 1 safariversion"
    ]
  }
}
```

Start-client-dev.js

```
// Do thisasthefirstthingsothatanycodereadingitknowstherightenv.
process.env.BABEL_ENV = 'development';
process.env.NODE_ENV = 'development';

// Makesthescriptcrashonunhandledrejectionsinsteadofsilently
// ignoringthem. In thefuture, promiserejectionsthatarenothandledwill
// terminatethe Node.js processwith a non-zeroexitcode.
process.on('unhandledRejection', (err) => {
  throwerr;
});

// Ensureenvironmentvariablesareread.
require('../config/env');

constfs = require('fs');
constchalk = require('react-dev-utils/chalk');
constwebpack = require('webpack');
constWebpackDevServer = require('webpack-dev-server');
constclearConsole = require('react-dev-utils/clearConsole');
constcheckRequiredFiles = require('react-dev-utils/checkRequiredFiles');
const {
  choosePort,
  createCompiler,
  prepareProxy,
  prepareUrls,
} = require('react-dev-utils/WebpackDevServerUtils');
constopenBrowser = require('react-dev-utils/openBrowser');
const{ checkBrowsers } = require('react-dev-utils/browsersHelper');
constsemver = require('semver');
constpaths = require('../config/constants/paths');
constconfigFactory = require('../config/webpack/webpack.config');
constcreateDevServerConfig = require('../config/webpack/webpackDevServer.config');
constgetClientEnvironment = require('../config/webpack/env');

// eslint-disable-next-lineimport/no-dynamic-require
constreact = require(require.resolve('react', { paths: [paths.appPath] }));

constenv = getClientEnvironment(paths.publicUrlOrPath.slice(0, -1));
constuseYarn = fs.existsSync(paths.yarnLockFile);
constisInteractive = process.stdout.isTTY;

// Warnandcrashifrequiredfilesaremissing
if(!checkRequiredFiles([paths.appHtml, paths.appIndexJs])) {
  process.exit(1);
}

// Tools like Cloud9 relyonthis.
const DEFAULT_PORT = parseInt(process.env.PORT, 10) || 3000;
const HOST = process.env.HOST || '0.0.0.0';

if (process.env.HOST) {
  console.log(
    chalk.cyan(
      `Attemptingtobindto HOST environmentvariable: ${chalk.yellow(
        chalk.bold(process.env.HOST)
      )}`
    )
  );
  console.log(
    'Ifthiswasunintentional, checkthatyouhaven\'t mistakenlysetitinyourshell.'
  );
  console.log(
    `Learnmorehere: ${chalk.yellow('https://cra.link/advanced-config')}`
  );
  console.log();
}

checkBrowsers(paths.appPath, isInteractive)
```

```
.then(() =>choosePort(HOST, DEFAULT_PORT))
.then((port) => {
  if (port == null) {
    // We havenotfound a port.
    return;
  }

  constconfig = configFactory('development');
  constprotocol = process.env.HTTPS === 'true' ? 'https' : 'http';
  // eslint-disable-next-lineimport/no-dynamic-require
  constappName = require(paths.appPackageJson).name;

  constuseTypeScript = fs.existsSync(paths.appTsConfig);
  consttscCompileOnError = process.env.TSC_COMPILE_ON_ERROR === 'true';
  consturls = prepareUrls(
    protocol,
    HOST,
    port,
    paths.publicUrlOrPath.slice(0, -1)
  );
  constdevSocket = {
    // eslint-disable-next-lineno-use-before-define
    warnings: (warnings) =>devServer.sockWrite(devServer.sockets, 'warnings', warnings),
    // eslint-disable-next-lineno-use-before-define
    errors: (errors) =>devServer.sockWrite(devServer.sockets, 'errors', errors),
  };
  // Create a webpackcompilerthatisconfiguredwithcustommessages.
  constcompiler = createCompiler({
    appName,
    config,
    devSocket,
    urls,
    useYarn,
    useTypeScript,
    tscCompileOnError,
    webpack,
  });
  // Loadproxyconfig
    // eslint-disable-next-lineimport/no-dynamic-require
  constproxySetting = require(paths.appPackageJson).proxy;
  constproxyConfig = prepareProxy(
    proxySetting,
    paths.appPublic,
    paths.publicUrlOrPath
  );
  // Servewebpackassetsgeneratedbythecompilerover a webserver.
  constserverConfig = createDevServerConfig(
    proxyConfig,
    urls.lanUrlForConfig
  );
  constdevServer = newWebpackDevServer(compiler, serverConfig);
  // Launch WebpackDevServer.
  devServer.listen(port, HOST, (err) => {
    if (err) {
      console.log(err);
      return;
    }
    if (isInteractive) {
      clearConsole();
    }

    if (env.raw.FAST_REFRESH&&semver.lt(react.version, '16.10.0')) {
      console.log(
        chalk.yellow(
          `Fast RefreshrequiresReact 16.10 orhigher. You areusingReact ${react.version}.`
        )
      );
    }

    console.log(chalk.cyan('Startingthedevelopmentserver...\n'));
    openBrowser(urls.localUrlForBrowser);
  });

  ['SIGINT', 'SIGTERM'].forEach((sig) => {
    process.on(sig, () => {
      devServer.close();
      process.exit();
    });
  });
```

```
    if (process.env.CI !== 'true') {
      // Gracefullyexitwhenstdinends
      process.stdin.on('end', () => {
        devServer.close();
        process.exit();
      });
    }
  })
  .catch((err) => {
    if (err&&err.message) {
      console.log(err.message);
    }
    process.exit(1);
  });
```

AddTarget.js

```
importReact, { useCallback, useEffect, useState } from "react";
import{ observer } from "mobx-react-lite";
import{ FaEdit, VscTrash } from "react-icons/all";
importAddTargetModalfrom"./AddTargetModal";
import{ useStores } from "../../stores/context";

constAddTargets = () => {
  const{ targetsStore } = useStores();
  const [showModal, setShowModal] = useState(false);
  const [editModal, setEditModal] = useState(false);
  const [clickedId, setClickedId] = useState(0);
  const [selectAccountModal, setSelectAccountModal] = useState(false);

  constfetchTargets = useCallback(async () => {
    awaittargetsStore.getAllTargets();
  }, [showModal]);

  consthandleRemoveTarget = useCallback(async (id) => {
    const r = window.confirm("Do youreallywanttodeletetarget?");
    if (r === true) {
      awaittargetsStore.destroy(id);
      awaitfetchTargets();
    }
  }, []);

  consttoggleModal = () =>setShowModal(!showModal);
  useEffect(fetchTargets, [fetchTargets]);
  return (
    <divclassName="add-targets-container">
      <divclassName="add-targets">
        <spanclassName="add-targets-header">Add Target</span>
        <divclassName="cards-container">
          <divclassName="cardadd" onClick={toggleModal}>
            <imgsrc="/images/add_client.svg" alt="addclient" />
            <span>Add New Target</span>
          </div>
          {targetsStore.targets.map((target) => (
            <divclassName="card" key={target.name}>
              <divclassName="actions-container">
                <FaEdit
                  onClick={() => {
                    setClickedId(target.id);
                    setEditModal(!editModal);
                  }}
                />
                <VscTrashonClick={() =>handleRemoveTarget(target.id)} />
              </div>
              {target.logo !== null ? (
                <divclassName="img-container">
                  <imgsrc={target.logo} alt="logo" />
                </div>
              ) : (
                <spanclassName="company-name">{target.name}</span>
              )}
            </div>
          ))}
        </div>
      </div>
      {showModal&& (
        <AddTargetModal
          toggleModal={toggleModal}
```

```
          setCreatedTargetId={setCreatedTargetId}
          toggleParsingModal={() =>setSelectAccountModal(!selectAccountModal)}
        />
      )}
      {editModal&& (
        <AddTargetModal
          toggleModal={() =>setEditModal(!editModal)}
          editable
          id={clickedId}
        />
      )}
    </div>
  );
};

exportdefaultobserver(AddTargets);
```

AddTargetModal.js

```
importReact, { useCallback, useEffect, useState } from 'react';
importPropTypesfrom 'prop-types';
import{ useParams } from 'react-router-dom';
import{ GrClose } from 'react-icons/all';
import{ observer } from 'mobx-react-lite';
import{ useStores } from '../../stores/context';

constAddProspectModal = (props) => {
  const{ clientId } = useParams();
  const{ companiesStore } = useStores();
  const [active, setActive] = useState(false);
  const [clientLogo, setClientLogo] = useState();

  const {
    toggleModal, id, editable, setCreatedProspectId, toggleParsingModal
  } = props;

  constfetchCompanyProspects = useCallback(async () => {
    companiesStore.getCompanyProspects(clientId);
  }, []);

  constsetEditable = useCallback(async () => {
    if (editable) {
      awaitcompaniesStore.setEditable({
        ...companiesStore.prospects.find((p) => p.id === parseInt(id, 10))
      });
    }
  }, [toggleModal, id, editable]);

  consthandleImage = async (event) => {
    setClientLogo(event.target.files[0]);
  };

  consthandleSubmit = async (event) => {
    event.preventDefault();
    constformData = newFormData();
    formData.append('clientLogo', clientLogo);
    formData.append('link', companiesStore.editable.link);
    formData.append('name', companiesStore.editable.name);
    formData.append('businessArea', companiesStore.editable.businessArea);
    formData.append('companyId', clientId);

    letres = {};

    if (companiesStore.isEditMode) {
      res = awaitcompaniesStore.update(companiesStore.editable.id, formData);
    } else {
      res = awaitcompaniesStore.create(formData);
      setCreatedProspectId(res.prospect.id);
      toggleParsingModal();
    }

    if (res.status === 200) {
      toggleModal();
      companiesStore.setEditable({});
    }
  };

  consthandleChange = useCallback((event) => {
```

```
    companiesStore.setEditable({
      ...companiesStore.editable, [event.target.id]: event.target.value,
    });
                                                if                          (companiesStore.editable.name
&&companiesStore.editable.link&&companiesStore.editable.businessArea) {
      setActive(true);
    } else {
      setActive(false);
    }
  }, []);

  useEffect(setEditable, [setEditable]);
  useEffect(fetchCompanyProspects, [fetchCompanyProspects]);

  return (
    <divclassName="add-prospect-modal">
      <divclassName="modal-container">
        <divclassName="create-client-profile-card">
          <divclassName="close-button">
            <GrCloseonClick={() => {
              toggleModal();
              companiesStore.setEditable({});
            }}
            />
          </div>
          <divclassName="header">
            ProspectCompany's Profile
          </div>
          <divclassName="form-group">
            <formonSubmit={handleSubmit} encType="multipart/form-data">
              <divclassName="general-information">
                <spanclassName="information-header">Generalinformation</span>
                <labelhtmlFor="name">Company Name</label>
                        <inputname="name"  id="name"  type="text"  placeholder="Enter  Company  Name"
onChange={handleChange} value={companiesStore.editable.name} required />
                <labelhtmlFor="link">Company'sLinkedinlink</label>
                  <inputname="link" id="link" type="text" placeholder="Enter Company'sLinkedinlink"
onChange={handleChange} value={companiesStore.editable.link} required />
                <labelhtmlFor="businessArea">Businessarea</label>
                <inputname="businessArea" id="businessArea" type="text" placeholder="Enter Company's
Business Area" onChange={handleChange} value={companiesStore.editable.businessArea} required />
                <labelhtmlFor="clientLogo">Client'slogo (optional)</label>
                  <inputname="clientLogo" id="clientLogo" className="file" type="file" accept=".jpg,
.jpeg, .png, .svg" onChange={handleImage} />
              </div>
              <divclassName="additional-information">
                <spanclassName="information-header">Additional Information</span>
                <labelhtmlFor="companyEmployees">ProspectCompany'sEmployees (optional)</label>
                <inputname="companyEmployees" id="companyEmployees" type="file" className="file" />
                <labelhtmlFor="otherFields">Someotherfields (optional)</label>
                <inputname="otherFields" id="otherFields" type="file" className="file" />
              </div>
               <buttontype="submit" className={active ? 'active' : ''}>{companiesStore.isEditMode ?
'Update Prospect' : 'AddProspect'}</button>
            </form>
          </div>
        </div>
      </div>
      <divclassName="modal-overlay" id="modal-overlay" />
    </div>
  );
};

AddProspectModal.propTypes = {
  toggleModal: PropTypes.func.isRequired,
};

exportdefaultobserver(AddProspectModal);
```

hooks/index.js

```
importqsfrom 'qs';
import{ useLocation } from 'react-router-dom';
import{ useMemo } from 'react';

// eslint-disable-next-lineimport/prefer-default-export
exportconstuseQueryParams = () => {
  const{ search } = useLocation();
  returnuseMemo(() =>qs.parse(search.replace('?', '')), [search]);
};
```

Target.js

```
import{ types } from "mobx-state-tree";

constSocialNetwork = types.model("SocialNetwork", {
  name: types.maybe(types.string),
  link: types.maybe(types.link),
});

constCourtDecision = types.model("CourtDecision", {
  court: types.maybe(types.string),
  date: types.maybe(types.date),
  number: types.maybe(types.number),
  type: types.maybe(types.string),
  case: types.maybe(types.string),
  form: types.maybe(types.string),
});

constAvailablePassword = types.model("AvailablePassword", {
  type: types.maybe(types.string),
  login: types.maybe(types.string),
  password: types.maybe(types.string),
});

const Target = types.model("Target", {
  id: types.maybe(types.identifierNumber),
  phoneNumber: types.optional(types.string, ""),
  email: types.optional(types.string, ""),
  passportDetails: types.optional(types.string, ""),
  socialNetworks: types.maybeNull(types.array(SocialNetwork)),
  courtDecisions: types.maybeNull(types.array(CourtDecision)),
  availablePasswords: types.maybeNull(types.array(AvailablePassword)),
  photos: types.optional(types.array(types.string)),
});

exportdefault Target;
```

session-store.js

```
import{ types } from 'mobx-state-tree';
import User from'../models/user';
importSessionStoreActionsfrom'./actions/session-store';

constSessionStore = types
  .model('SessionStore', {
    user: types.maybeNull(User),
  })
  .views((self) => ({
    getisAuthenticated() {
      returnBoolean(self.user);
    },
  })).actions(SessionStoreActions);

exportdefaultSessionStore;
```

context.js

```
import{ createContext, useContext } from 'react';

constRootStoreContext = createContext({});
exportconstRootStoreProvider = RootStoreContext.Provider;
exportconstuseStores = () =>useContext(RootStoreContext);
```

root-store.js

```
import{ flow } from 'mobx-state-tree';

constRootStoreActions = (self) => ({
  restore: flow(function* restore() {
    yieldself.sessionStore.restore();
  }),
});

exportdefaultRootStoreActions;
```

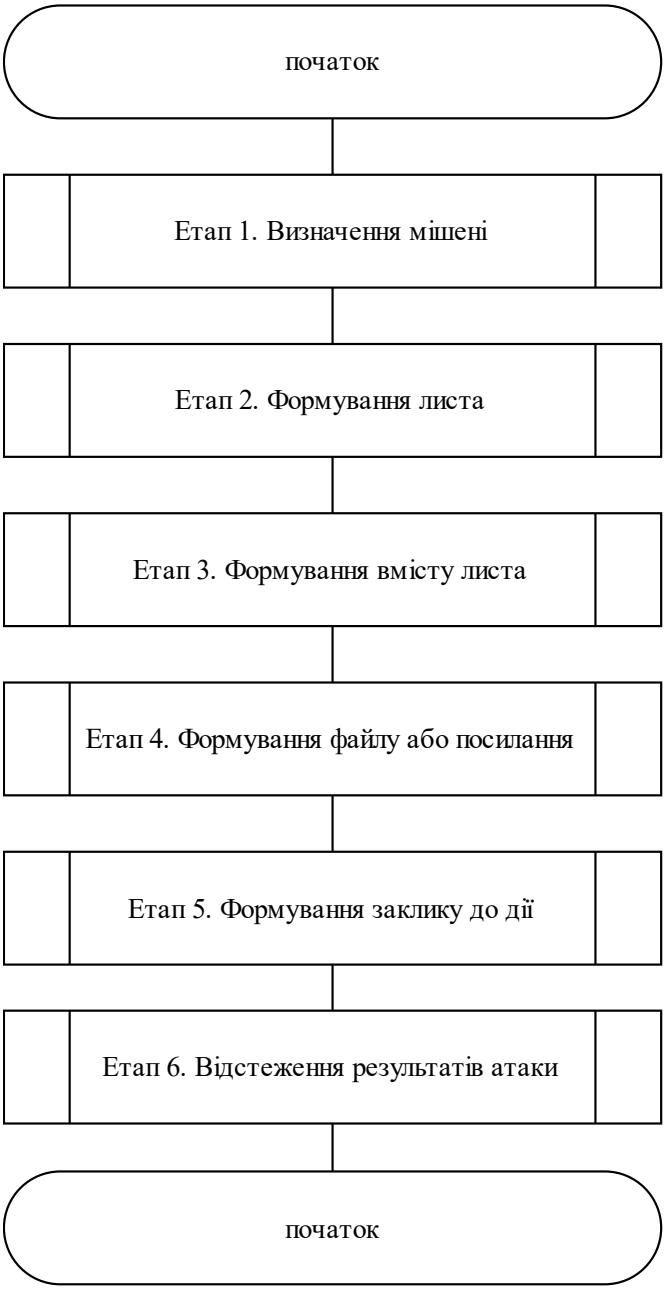
index.jsx

```
importReactfrom 'react';
importReactDOMfrom 'react-dom';
import './index.css';
importAppfrom'./App';
ReactDOM.render(<App />, document.getElementById('root'));
```

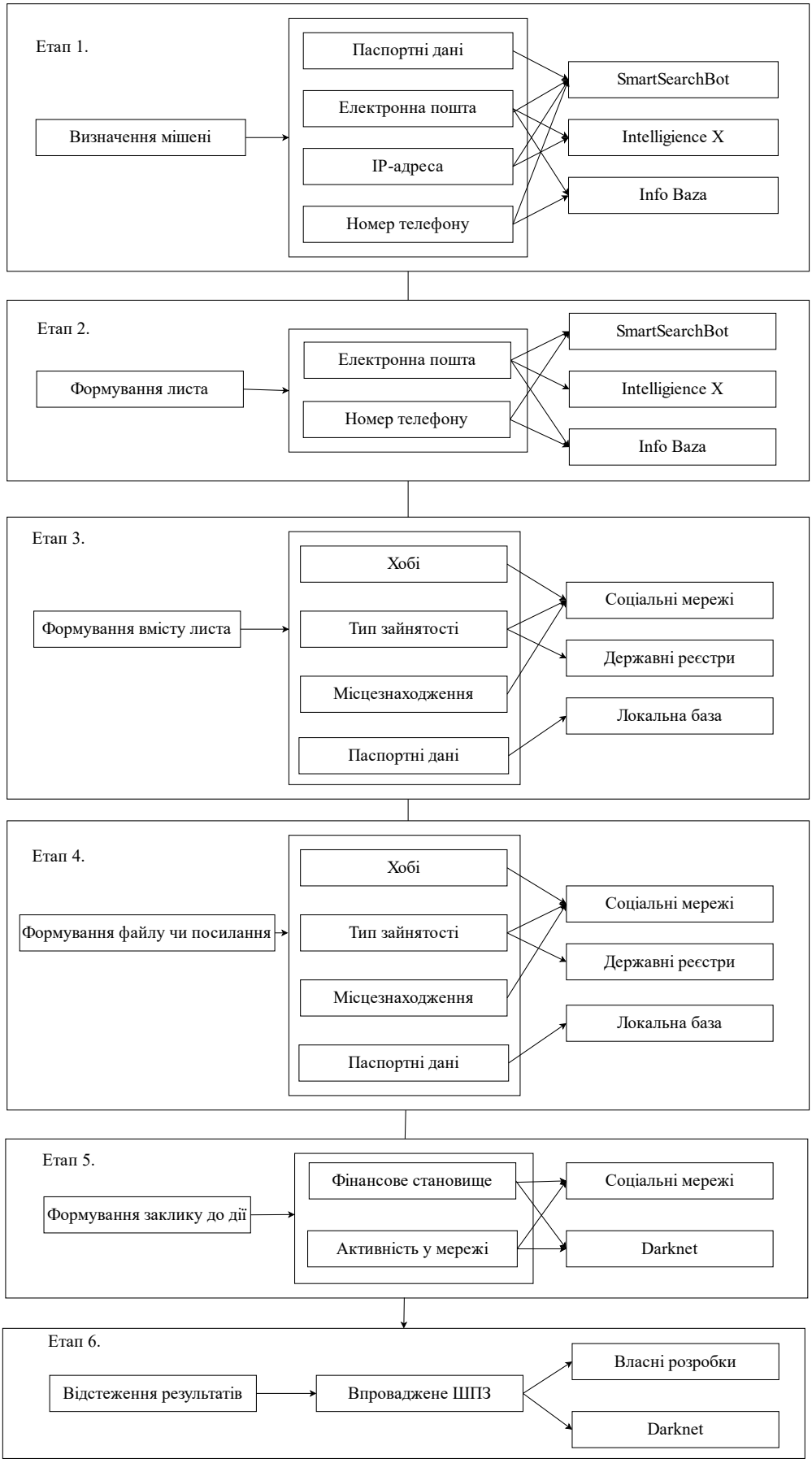
ІЛЮСТРАТИВНА ЧАСТИНА

до магістерської кваліфікаційної роботи
на тему:
МЕТОД ТА ЗАСІБ ЗБИРАННЯ ІНФОРМАЦІЇ ПРО МІШЕНЬ ПІД ЧАС ІНФОРМАЦІЙНОЇ ВІЙНИ

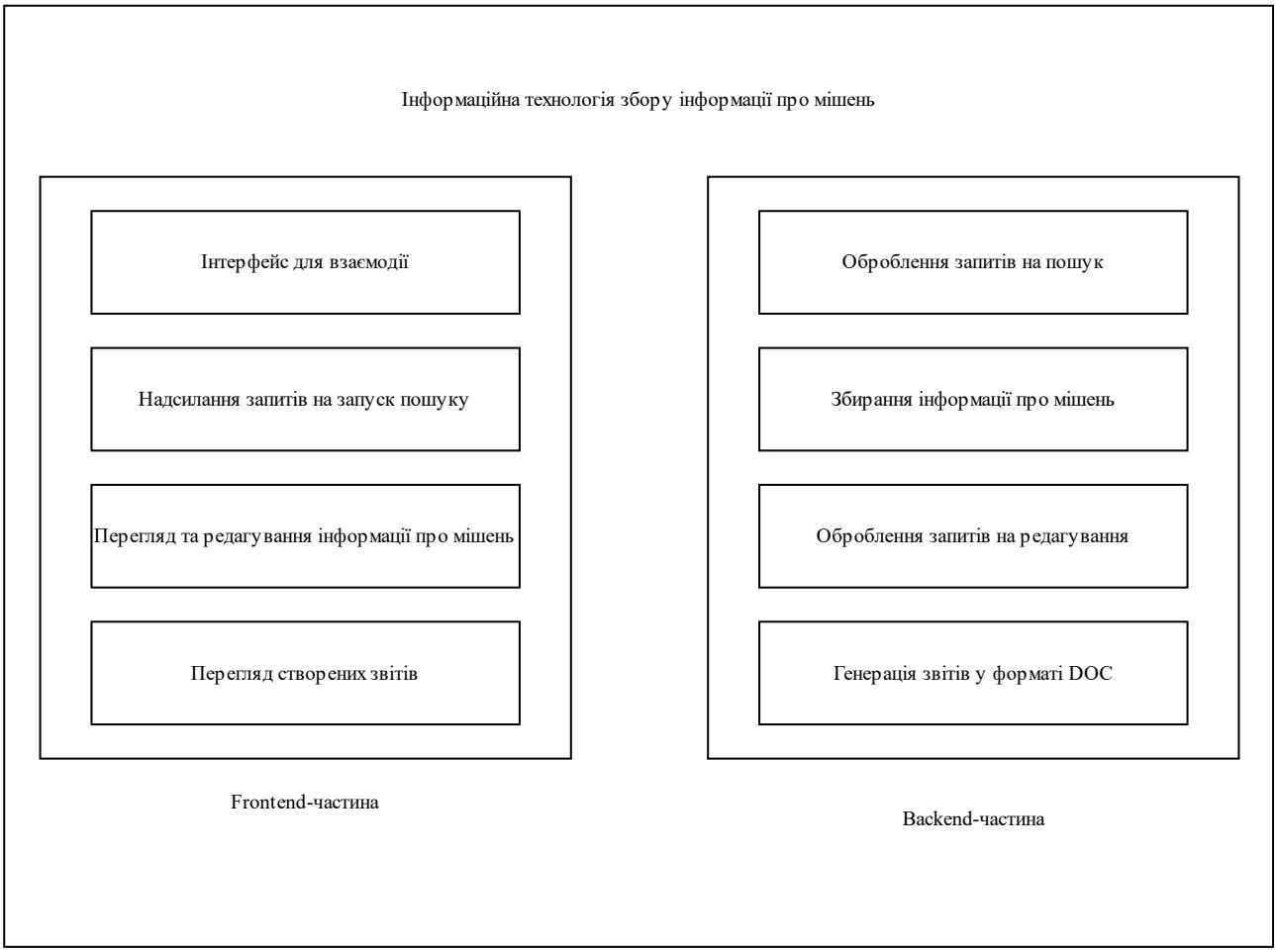
ЕТАПИ ПРОВЕДЕННЯ АТАКИ



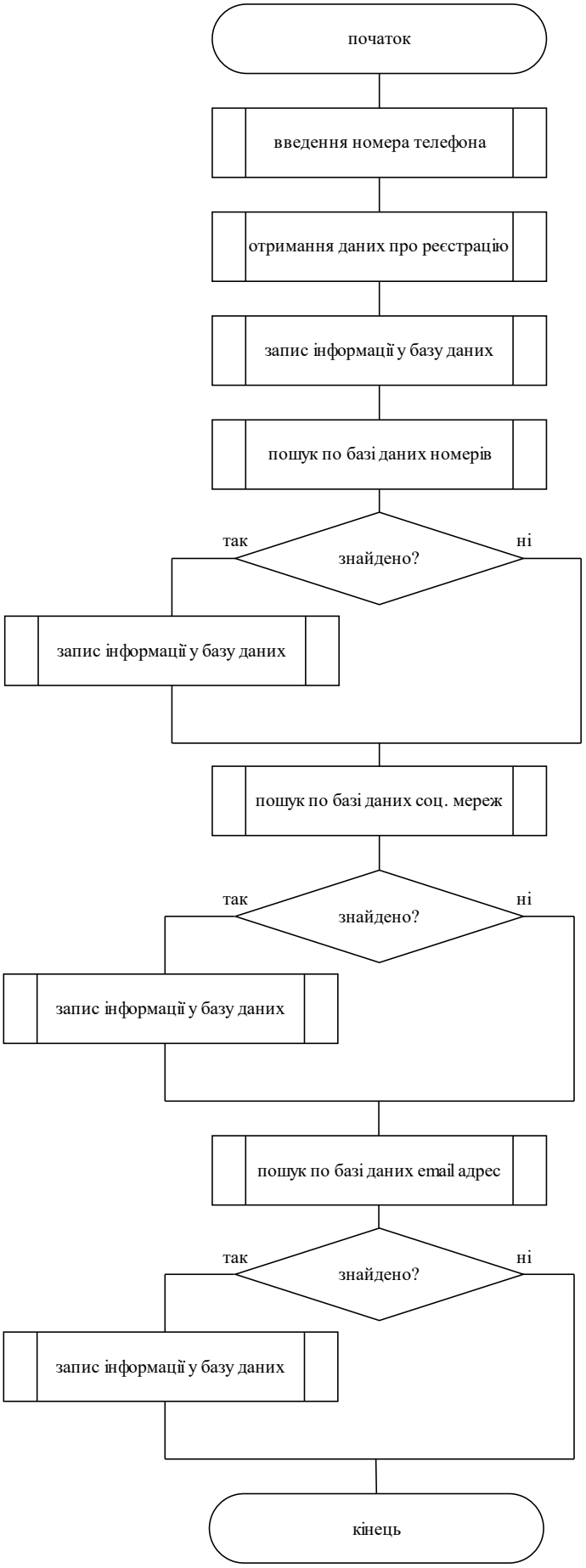
МОДЕЛЬ ПРОВЕДЕННЯ АТАК



СИСТЕМНІ МОДУЛІ



АЛГОРИТМ ПРОЦЕСУ ЗБОРУ ІНФОРМАЦІЇ



ЗАГАЛЬНА СХЕМА РОБОТИ СИСТЕМИ

РЕЗУЛЬТАТИ ТЕСТУВАННЯ