МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____Ю. П. Кондратенко

«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ БОТ-АТАК НА ОСНОВІ АНАЛИЗУ КОРИСТУВАЦЬКИХ ПРОФІЛЕЙ У СОЦІАЛЬНИХ МЕРЕЖАХ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810316

Виконав студент 4-го курсу, групи 402

_____ ***Є. А. Кучеренко***

«___» червня 2022 р.

Керівник: професор

_____***І. М. Журавська***

«___» червня 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**

Спеціальність **122 «Комп'ютерні науки»**

(шифр і назва)

Галузь знань **12 «Інформаційні технології»**

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____ Ю. П. Кондратенко

«____» _____20___р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Кучеренко Єгору Андрійовичу.

1. Тема кваліфікаційної роботи «Автоматизована система виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах».

Керівник роботи Журавська Ірина Миколаївна, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____ 20___ р. № ____

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: вебпосилання на користувацький профіль у соціальній мережі «Facebook».

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз методів та систем основ збору та аналізу інформації з відкритих джерел;

– принципи вибору метрик та системи оцінки профілю користувача;

– визначення та обгрунтування основних принципів та методик роботи з процесами збору інформації у порівняльній характеристиці;

– демонстрація отриманого набору даних з пов’язаних між собою профілів користувачів.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Аналіз вимог до умов праці, заходи з техніки безпеки та охорона праці під час роботи за комп’ютером»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	**Ст. викладач кафедри екології, канд. техн. наук Алексєєва А. О.**	

Керівник роботи __д-р техн. наук, професор Журавська І. М._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Кучеренко Є. А._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання «_ _» _____ 202___ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема :__Автоматизована система виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	10.11.2021	15.11.2021	Виконано
2	Отримання завдання на виконання БКР	21.11.2021	21.11.2021	Виконано
3	Складання календарного плану	05.12.2021	07.12.2021	Виконано
4	Початок виконання БКР: збір матеріалу та даних, написання аналітичної частини БКР	28.02.2022	27.03.2022	Виконано
5	Отримання завдання на переддипломну практику	20.05.2022	20.05.2022	Виконано
6	Проходження переддипломної практики, написання практичної частини БКР	23.05.2022	05.06.2022	Виконано
7	Розробка ПЗ для БКР	25.05.2022	03.06.2022	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	31.05.2022	31.05.2022	Виконано
9	Розробка звіту з переддипломної практики	01.06.2022	03.06.2022	Виконано
10	Доробка та остаточне оформлення БКР	03.06.2022	19.06.2022	Виконано
11	Подання БКР рецензенту	16.06.2022	19.06.2022	Виконано
12	Подання БКР, електронні копії та інші необхідні документи до захисту	20.06.2022	20.06.2022	
13	Захист БКР перед екзаменаційною комісією	27.06.2022	27.06.2022	

Розробив студент _____Кучеренко Є. А _____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи _д-р техн. наук, професор Журавська І. М._ _____

(наук. ступінь, вчене звання, прізвище та ініціали) (підпис)

«____» _____ 2021 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента 402 групи
ЧНУ ім. Петра Могили

Кучеренко Єгора Андрійовича

Тема: «Автоматизована система виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах»

Мета та основні результати роботи: розробка та створення автоматизованої системи протидії flame&flood-атакам на підставі збору і аналізу даних з соціальних мереж. Розроблена система здійснює автоматичний обходу та вилучення фрагментів інформації з вебсторінок з подальшим процесом генерування графічного представлення взаємозв'язків отриманих даних.

У першому розділі представлено огляд способів протидії flame&flood-атакам за допомогою автоматизованого аналізу профілів соціальних інтернет-мереж відповідно до заданого набору параметрів для можливого виявлення потенційних учасників бот-мережі та попередження інших учасників про виявлену загрозу маніпулювання їх соціальною поведінкою.

Другий розділ включає в себе опис технологій збору та аналізу інформації, бібліотек та їх переваг і недоліків, наприклад, за допомогою технології «парсинг» – автоматизованого аналізу сирцевого коду вебсторінки. За такою технологією можливо відокремлення певних елементів вебсторінки та отримання їх значень. Наприклад, це може бути пошук HTML-тегу *<title>* та отримання назви вебсторінки або будь-якого іншого тегу та елементу вебсторінки і його значення.

У третьому розділі наведено детальний опис внутрішніх механізмів розробленої автоматизованої системи та способів їх взаємодії і функціонування, а саме наведено взаємозв'язки між функціями сирцевого коду та їх детальних опис.

У спеціальній частині, присвяченій охороні праці, наведено основні засоби та заходи безпеки для працівників у офісних приміщення відповідно до ДСТУ.

Практична значимість розробленої системи полягає у протидії flood&flame-атакам з метою перешкоджання та запобігання гострим соціально-політичним конфліктам між двома чи більшою кількістю груп на просторах соціальних мереж.

Сторінок – 68 (без додатків), табл. – 6, рис. – 27, додатків – 2, джерел посилання – 20.

Ключові слова: бот-атака, соціальна мережа, користувацький профіль, парсинг, скрапінг, краулінг, інформаційний пошук, автоматизована система.

ABSTRACT

of the Bachelor's Thesis

"Automatized bot attack detection system based on the analysis of the user profiles on social networks"

Student of group 402: Kucherenko Yehor Andriiovych

Supervisor: Dr. Sc. (Eng.), Professor Zhuravska I. M.

Purpose and main results of the work: development and creation of an automated system to counter flame & flood attacks based on the collection and analysis of data from social networks. The developed system automatically traverses and extracts fragments of information from web pages, followed by the process of generating a graphical representation of the relationships of the data.

The first section provides an overview of ways to counter flame & flood attacks using automated analysis of social network profiles for a given set of parameters to identify potential participants in the bot system and warn other participants about the threat of manipulating their social behavior.

The second section includes a description of information collection and analysis technologies, libraries, and their advantages and disadvantages – for example, using "parsing" technology – automated analysis of the raw code of a web page. With this technology, it is possible to separate certain elements of the web page and obtain their values. For example, this could be finding the HTML <title> tag and getting the name of the webpage or any other tag and element of the webpage and its meaning.

The third section provides a detailed description of the internal mechanisms of the system and ways of their interaction and functioning, namely – the relationship between the functions of the raw code and their detailed description.

In the special part devoted to labor protection namely the basic means and safety measures for employees in office premises according to norms of DSTU are resulted.

The practical significance of the developed system is to counter flood & flame attacks to prevent and prevent acute socio-political conflicts between two or more groups on social networks.

Pages – 68 (without appendices), tables – 6, figures – 27, appendices – 2, references – 20.

Keywords: bot attack, social network, user profile, parsing, scraping, crawling, information search, automatized system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВДТ – візуальний дисплейний термінал
ЕОМ – електронна обчислювальна машина
КПО – коефіцієнт природної освітленості
ОС – операційна система
ПЕОМ – персональні електронні обчислювальні машини
ПЗ – програмне забезпечення
ПК – персональний комп’ютер
ССБП – система стандартів безпеки праці
ЦП – центральний процесор
ШНМ – штучна нейронна мережа

API – Application Program Interface
CIL – Common Intermediate Language
CLR – Common Language Runtime
GC – Garbage Collector
JSON – JavaScript Object Notation
JVM – Java Virtual Machine
JWT – JSON Web Token
MitM – Man-in-the-Middle
NLP – Natural Language Processing
SoC – System on Crystal
TFPS – Tera FLoating-point Operations Per Second
URI – Unique Resource Identifier
VE – Virtual Environment
WWW – World Wide Web

Пояснювальна записка

до кваліфікаційної роботи

на тему:

«АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ БОТ-АТАК НА ОСНОВІ АНАЛИЗУ КОРИСТУВАЦЬКИХ ПРОФІЛЕЙ У СОЦІАЛЬНИХ МЕРЕЖАХ»

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 402. 21810316

Виконав студент 4-го курсу, групи 402

_____ ***Є. А. Кучеренко***

«____» червня 2022 р.

Керівник: д-р техн. наук, професор

_____ ***І. М. Журавська***

«____» червня 2022 р.

Миколаїв – 2022

ВСТУП

Починаючи з 1983 р., з появою системи телекомунікації у рамках розвитку проєкту ARPANET, що створена за дорученням Міністерства оборони США, почалося дослідження та системний розвиток засобів передачі інформації між контрольними вузлами зв’язку, які були поєднані у загальну мережу. З плином часу та появою стандартів комунікаційних мереж, зокрема таких інститутів стандартизації, як ISO, Інститут інженерів з електроніки та електротехніки (IEEE), ITU (Міжнародний комітет з консультації з питань телефонії та телеграфії), ETSI (Європейський інститут телекомунікаційних стандартів) та згодом – IAB (Рада з архітектури Інтернету) – послуги зв’язку, зокрема глобальної мережі Інтернет, яка і з’явилася завдяки стандартизації та об’єднанню багатьох мереж різного рівня, стали доступними спочатку для профільних невійськових організацій, таких як школи та університети, потім для великих та середніх підприємств, а згодом – й для пересічних громадян.

Ц етапах розвитку мереж зв’язку, а зокрема мережі Інтернет – можна виділити декілька ключових моментів та періодів, наприклад, – з середини 1990-х до кінця 2010-х років, – коли відбулися не тільки бурхливий розвиток технології WWW, яку створив Сер Тімоті Джон Бернерс-Лі, а й значні зміни у соціально-економічній будові суспільства, які й дозволили укорінити ставлення до даної технології як до звичайної побутової потреби, на рівні з електро- та газопостачанням, опаленням.

Після низки еволюційних процесів у веброзробці та пропускній спроможності комп'ютерних мереж люди ще більше змінили своє ставлення до даної технології у соціальному аспекті, що дозволило частину їх повсякденної соціальної взаємодії та спілкування перенести у цифровий простір, зокрема на вебсайти та форуми, а згодом і у спеціальні середовища, які й отримали свою назву відповідно до соціального явища, які вони представляють – «соціальні мережі».

Взаємодія окремих людей, груп та представників цілих організацій у просторі соціальних мереж, за умови, що кожна взаємодія та діалог є задокументованими постфактум або у самій мережі, або при виконанні процесу логування з боку вебсервера – спричинили появу нових видів поведінки та соціальної взаємодії, до яких також можна віднести й «соціальні атаки» – вид взаємодії, спрямований на нанесення матеріальної чи психічної шкоди певному суб'єкту [12].

Виконання аналізу таких видів поведінки за допомогою сучасних засобів автоматизованих обчислень, заснованих на алгоритмах збору інформації з відкритих джерел, надасть змогу не тільки долучити нові знання про людину як соціальну істоту, але й спрогнозувати її поведінку та перешкодити появі таких видів атак на просторах соціальних мереж або інших систем масових комунікації.

Метою роботи є розробка системи для автоматизованого виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах.

Об'єктом дослідження є процес автоматизації систем для аналізу структурованої інформації, отриманої з відкритих джерел.

Предметом дослідження є методи програмної та алгоритмічної реалізації цілісної системи для виконання автоматизованого пошуку, перевірки й формування висновків щодо здійснення бот-атак для кожного користувацького профіля у певній соціальній мережі.

Для досягнення мети, необхідно виконати наступі завдання:

– виконати аналіз типів та методів доступу до інформації з користувацьких профілів шляхом автоматизації;

– виконати аналіз систем захисту збору інформації о користувачах з боку соціальних мереж;

– визначити основні напрямки розробки та певний інструментальний набір технічних засобів для виконання збору та аналізу інформації, зокрема: мови програмування, операційної системи та програмного оточення системи, що

розробляється, набір допоміжних зовнішніх бібліотек для прискорення процесу розробки;

– розробити загальний алгоритм обходу дерева вебсторінок усередині соціальної мережі;

– розробити основні принципи взаємодії елементів екосистеми збору інформації, а саме – процесу взаємодії технік скрапінгу, парсингу та вебкраулінгу.

Методологія і методи досліджень. Для досягнення поставленої мети використовуються технології мов програмування, які належать до класу інтерпретуємих, а саме – мови програмування Python, програмні бібліотеки сирцевого коду для взаємодії з вебсерверами та виконання вебзапитів та система побудови елементів користувацького графічного інтерфейсу і математичних графіків.

Апробація результатів дипломної роботи відбувалася на базі матеріально-технічного забезпечення Чорноморського національного університету імені Петра Могили з використанням обчислювальних потужностей частини комп'ютерної техніки університету. Результати дипломної роботи були представлені під час Всеукраїнської науково-практичної конференції «Інформаційні технології та інженерія» в ЧНУ ім. Петра Могили (Миколаїв, 8–11 лютого 2022 р.). За результатами бакалаврської роботи опубліковано 1 друковану працю [10].

Практична значимість розробленої системи полягає у тому, що за її допомогою можливо виконувати пошук та передчасне знешкодження ворожих бот-систем, які спричиняють матеріально-технічні та соціальні збитки.

Дипломна робота складається зі вступу, 4 розділів, висновків, переліку джерел посилання, додатків. Загальний обсяг фахової частини роботи складає 68 сторінок (без додатків), 6 табл., 27 рис., 2 додатки та 20 джерел посилання на літературні джерела.

1 МЕТОДИ ТА ЗАСОБИ ЗАХИСТУ ВІД FLAME&FLOOD-АТАК НА ПІДСТАВІ АНАЛІЗУ ТЕХНОЛОГІЙ ЗБОРУ ІНФОРМАЦІЇ З ВІДКРИТИХ ДЖЕРЕЛ

1.1 Аналіз предметного середовища та основних технологій збору інформації

Інформація – це певна міра визначеності щодо об’єкта або суб’єкта, або іншими словами – це міра ентропії. Ентропія у теорії інформації – це міра невизначеності системи у певний момент часу. Тобто, чим більше є інформації про систему, ти меншою є її інформаційна ентропія. Будь яку інформацію можна умовно поділити на інформацію з першоджерела та вторинну інформацію. Інформація з першоджерела. Або первинна інформація – це така інформація, що була згенерована системою (штучною або природньою) та є унікальною, тобто такою, яка ще не зустрічалась у полі простору усіх можливих комбінації мінімальних часток інформації – бітів.

Основними технологіями збору інформації є парсинг, скрапінг та вебкраулінг у середовищі мережі Інтернет та всесвітньої вебпавутини.

З розвитком технічного прогресу та систем зв’язку, зберігання та пошуку інформацій у загальному інформаційному просторі збільшилась загальна кількість інформації, що генерується системами та людиною, але відносний відсоток унікальної інформації, у сенсі зазначеному вище, змінився несуттєво. Основною причиною цього стали два фактори:

а) загальна розповсюдженість наукових посилань та синтез нових наукових робіт та досліджень на підставі існуючих або попередніх, попереднім фактором до цього стала можливість маже миттєвого обміну знаннями та напрацюваннями за допомогою глобальних мереж зв’язку та комунікацій різних направленостей та методів передачі у масштабах планети. З іншого боку, саме це призвело до

стрімкого зростання відсотку цитування інших робіт – тобто вкраплення існуючої інформації та генерації нової на її підставі, що зменшує загальний відсоток унікальної, нової інформації. Звісно можна зауважити, що будь-яка інформація, за рідкісним виключенням, наприклад, нові наукові відкриття у галузі, що тільки почала розвиватися, не є абсолютно новими чи унікальними – але в даному випадку йде мова саме про питання у більш загальному сенсі;

б) значне полегшення доступу до інформації та спрощення і здешевлення матеріально-апаратного забезпечення, а саме:

1) значне зростання обчислювальних можливостей центрального процесора (ЦП) – збільшення показника TFPS (терафлопс) та збільшення щільності кількості транзистор на кристалі за законом Мура, появи SoC (System on Crystal), появи нових апаратних та програмних архітектури – X64, ARM та систем конвеєрних задач у середні регістрів ЦП і їх розподілення;

2) стрімкий розвиток мережі Інтернет – поява стандартів Wi-Fi (802.11), WiMAX (802.16d) для використання у мережах WLAN, WMAN;

3) поступова мініатюризація та персоналізація пристроїв – тобто поява КПК, планшетних комп’ютерів, а згодом і смартфонів;

4) загальне здешевлення та стандартизація виробництва електронних компонентів, перехід до конвеєрної, розподіленої системи виробництва, кожен з етапів якого спеціалізується на певному підвиді обладнання та комплектуючих для фінального продукту.

Все це призвело до бурхливого зростання використання мережі Інтернет та мережі вебсторінок – WWW – пересічними користувачами, які до цього не мали можливості їх використання, або використовували їх у обмеженому вигляді на підприємствах або під час / задля вирішення робочих питань та як елемент робочого процесу. Збільшення користувачів мережі призвело до явища, яке називають «інформаційний вибух», – тобто до такого явища, коли кількість нових

даних та похідних від них зростає з експоненціальною швидкістю. Таке різке збільшення даних, через деякий час призвело не тільки до швидкого економічно-технічного прогресу через швидкій обмін даними, але й до змін у соціально-політичному становищі та соціальних норм поведінки користувачів мережі, що безпосередньо вплинуло на їх спосіб життя, наприклад:

1) поступове звикання населення до простого та швидкого доступу до будь-якої світової інформації за допомогою створених пошукових систем для індексації вебсторінок;

2) перехід багатьох компаній та перенесення їх серверної інфраструктури до систем розподілених виділених віртуальних серверів (VPS,VDS) та використання технології «Системи доставки контенту» або CDN, використання якої ще більше збільшує пропускну здатність мережі та зменшує загальний час затримки. Ці фактори «привчили» багатьох користувачів до думки, що їхні данні та файлі захищенні та доступні їм для використання з будь-якої точки земної кули, де є мережа Інтернет;

3) ефект новизни таких явищ, як Інтернет та електроні соціальні мережі, призвів до їх необґрунтованого використання, а саме до неповного розуміння користувачами механізмів роботи цих сервісів та важливості їх персональних даних та найважливіше – це відсутність розуміння, які саме дані користувачі передають та завантажують на віддалені сервери компаній власників соціальних мереж. Наприклад, при завантаженні фотографії користувач передає не тільки фактичну інформацію, яка описує саму фотографію – тобто кодування кожного пікселю, – а ще й метадані, в яких може зберігатися наступна інформація: тип та модель камери, на яку було зроблено фото, дата та час фотозйомки та навіть геодані, де саме було зроблено знімок.

Зазначені приклади соціальних явищ поступового впровадження технологій та, зокрема, соціальних мереж призвели до розповсюдження нових джерел ЗМІ,

якими тепер могли бути не тільки профільні та перевіренні видання, а будь-хто з користувачів мережі. Наслідками поширення такого формату висловлювання, орієнтованого на широку аудиторію стали:

1) збільшення кількості думок та точок зору на одну й ту саму ситуацію у інформаційному просторі;

2) формування соціальних груп та формувань людей з певною точкою зору – тобто розділення суспільства на певні ідеологічні та соціальні групи, не тільки в масштабах глобальних політичних, геополітичних та соціальних питань, як це було раніше, до провадження масових комунікацій, але й тепер в більш дрібних питаннях, що веде до прогресуючої сегментації суспільства.

Зазначений ефект сегментації може чинити як позитивний так і негативний вплив – тобто, як прискорювати соціальну еволюцію, шляхом все активнішої взаємодії все більш суперечливих підгруп, так і спричиняти все більш масові розлади у суспільстві, як природнім шляхом, тобто просто при не співпадінні думок та точок зору, так і провокуватись штучно – іншими соціальними групами задля досягнення певних власних інтересів.

Основою засобом проведення flood-атаки є багаторазове та масове відправлення повідомлень, які можуть мати безглуздий, агресивний або ще будь-який характер, що призводить до потрібної поведінки жертви та у решті-решт до мети, яку обрав зловмисник. Це може бути, наприклад, доведення цілі до стану психозу або іншого психічного розладу у легкій формі з метою шантажу або спричинення шкоди ще й фізичному здоров'ю жертви – наприклад, доведення до самогубства.

Основним ознаками flood-атаки є:

1) велика кількість повідомлень від інших учасників, з певної теми за короткий проміжок часу;

2) масове співпадіння тексту повідомлень та/або думок «людей», що коментують вислів жертви атаки;

3) мізерний словниковий запас тих , хто коментує вислів, наявність слів, що повторюються, неправильне використання частини мови та/або її неправильне відмінювання;

4) циклічне наведення одних й тих самих аргументів, якщо має місце певна суперечка;

5) масова поява акаунтів користувачів певної соціальної верстви, статі та віку, що коментують зазначені публікації;

6) отримання повідомлень та/або коментарів від користувацьких профілів, які завжди присутні у тих самих соціальних групах та групах з інтересів, що й ціль, яку треба атакувати, навіть, якщо ці групи дуже різні за тематикою та не відповідають соціальному статусу, статі та/або віку профілю користувача, який надсилає повідомлення жертві.

Flame-атаки, на відміну від flood-атак, можуть та найчастіше є більш короткочасними, бо мають інший механізм соціального впливу на обрану ціль. Головна ціль flame-атаки – виявлення «гострих» політичних та/або соціальних тем, а також соціальної групи (друзів, родичів, знайомих, колег тощо), до якої входить жертва даного виду атаки [3]. Крім того, метою flame-атаки є формування двох або більше суперечливих думок або складної – з соціально-політичного боку – тези чи висловлювання, що призводить до бурхливого обговорення обраної теми серед учасників соціальної групи. В результаті це може призвести до розпаду такої групи або погіршенню відносин серед її учасників. Варто зазначити, що на відміну від flood-атаки, основною метою flame-атаки є не формування певної думки за допомогою її масового повторення від «різних» джерел. Її головне призначення – це саме першочергове підняття «гострої» теми з її подальшим обговоренням самими учасниками обраної соціальної групи.

Основні ознаки flame-атаки:

– раптове підняття гострої соціально-політичної теми серед загального потоку повідомлень, який не має відношень до цих тем;

– намагання учасника групи звести тему розмови до певних, потрібних йому тем;

– публікація повідомлень, провокуючого та неоднозначного характеру, які можуть призвести до суперечки;

– зменшення активності учасника групи, який спровокував обговорення певної теми, після початку обговорення.

Найчастіше описані вище атаки виконуються за допомогою соціальних інтернет-мереж та системи бот-програм – спеціального виду автоматизованого програмного забезпечення, яке імітує соціальну активність реальних користувачів з першочерговою метою викликати довіру та емпатію інших учасників. Такий підхід призводить до більш зацікавленого обговорення теми з їх боку та формування певної думки за допомогою її циклічного повторення від різних екземплярів бот-кластеру [4].

Задля запобігання даним видам атак існує три основних методи:

1) перевірка профілю кожного з майбутніх учасників групи перед його вступом у цю групу – пре-модерація учасників у ручному режимі;

2) перевірка активності користувача у групі – модерація його постів та загальної активності на предмет розповсюдження flame&flood-повідомлень;

3) створення автоматичної системи збору інформації та статистики за кожним користувацьким профілем з подальшою метою аналізу активності.

Саме останній варіант і буде розглянуто у даній роботі.

1.2 Аналіз властивостей основних технологій збору інформації

Визначимо поняття «система». Система – це сукупність окремих об’єктів, які мають певний стан і функціональне призначення, сполучені між собою зв’язками задля обміну інформації чи іншої взаємодії та виконують певну роботу, тобто виконують якісь якісні та/або кількісні зміни задля досягнення певної мети, до якої прагне система.

Перш за все, відокремимо основні етапи, які будуть основою системи, що розроблюється.

Головним призначенням, системи що розробляється, є збір інформації про профіль користувача – тобто пошук та зберігання відкритої інформації, яку надає сам користувач про себе.

Перерахуємо та опишемо основні терміни, які використовуються при виконанні задачі збору та обробки інформації як окремі підвиди вирішення даної проблеми:

- скрапінг;
- парсинг;
- краулінг.

Використаємо графічну схему задля кращого розуміння ситуацій, у яких необхідно використовувати кожен з наведених методів (рис. 1.1).

Розглянемо вищенаведені поняття більш детально.

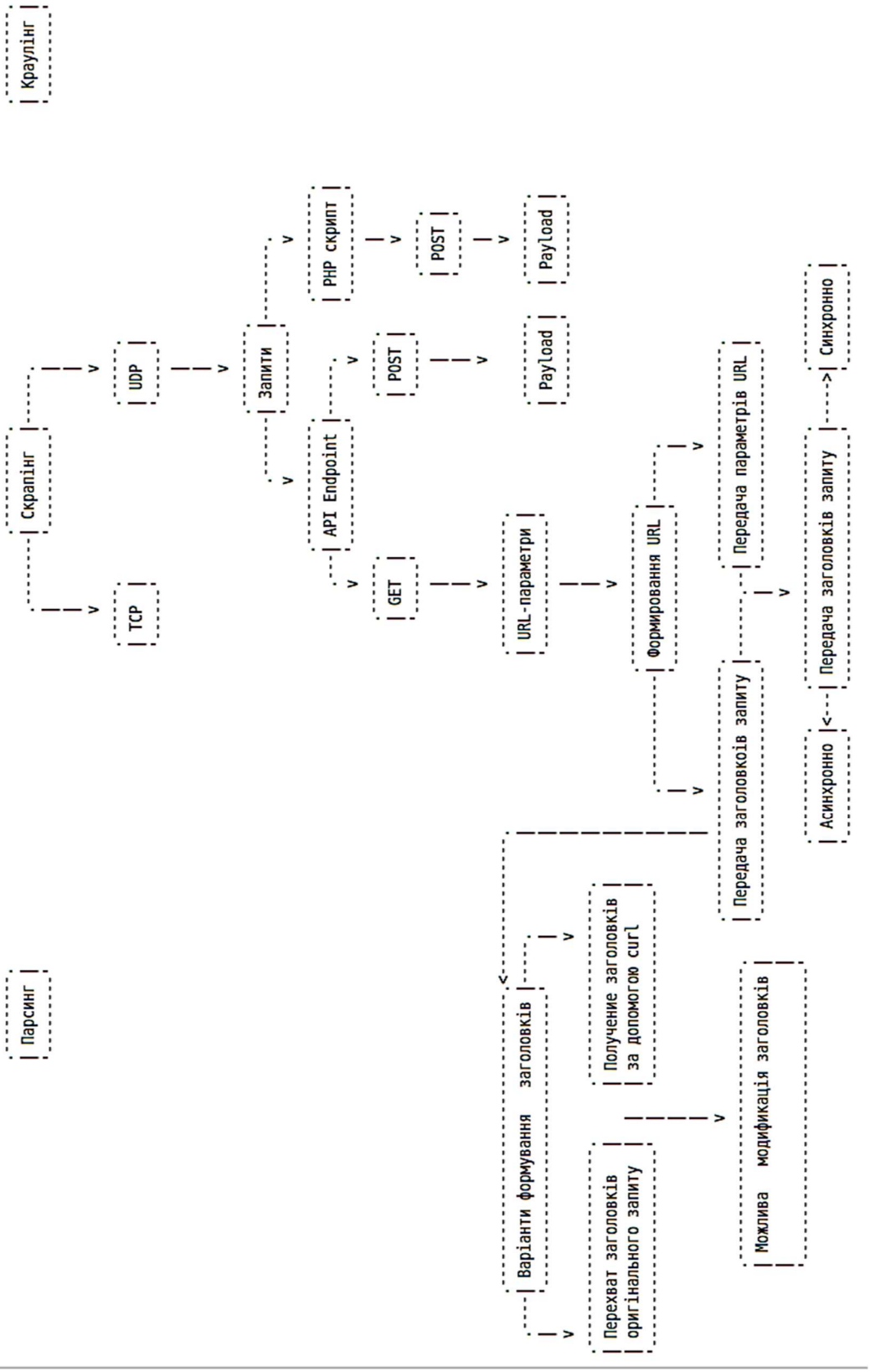


Рисунок 1.1 – Графічна схема доцільності використання методів збору інформації

1.2.1 Скрапінг

Скрапінг – це процес попереднього аналізу запитів до веб серверу з метою визначення структури та параметрів запиту для їх подальшого відтворення та отримання відповіді від вебсерверу з необхідними даними, задля яких було зроблено запит.

Процес вебскрапінгу підлягає частковій автоматизації, адже потребує присутності людини, яка буде виконувати первинний аналіз запитів, підбираючи засоби обходу захисту від несанкціонованого доступу.

Вебскрапінг не є різновидом вебатак, наприклад, таких, як атака «Людина посередині» або, як її ще називають, MITM-атака (Man-in-the-Middle). Розглянемо це питання більш детально. Вебскрапінг не є MITM-атакою через те, що при реалізації даного виду атак основним її вектором для першочергового доступу до жертви є перехоплення мережевого трафіку який йде від хоста жертви до цільового вебсерверу. Перехоплення вебтрафіку може бути реалізоване у два етапи, другий з яких полягає у використанні proxy-серверу. Перший етап атаки може бути реалізований за допомогою техніки, відомої як «спуфінг». Сам спуфінг – це ситуація, коли певна людина, система чи програмне забезпечення видає себе за іншу особу, систему чи програмне забезпечення. Важливо відмітити, що процес спуфінгу у середовищі програмного забезпечення не варто плутати з «троянським програмним забезпеченням», адже вони мають різні механізми взаємодії з користувачем та мають різне призначення.

При використанні троянського програмного забезпечення основою механізмів його роботи та поведінки – як і у механізмів спуфінгу – є часткова чи повна імітація діяльності ПЗ, під яке маскується шкідлива програма. Але спуфінг – це атака реального часу, тобто призначена для використання у безпосередній момент проведення атаки (наприклад DNS- чи DHCP-спуфінг) та перенаправлення користувача на інший ресурс, що належить зловмиснику. «Троянський кінь» у свою чергу використовує відкладене виконання певних дій, заданих розробником чи оператором (якщо «троянський кінь» поєднується з ще одним видом шкідливого

ПЗ – програмами віддаленого керування або використовуватись у якості оболонки для доставки іншого виду шкідливого ПЗ – програм для завантаження).

Тобто, основними елементами MitM-атаки є наявність проксі-серверу (proxy-server) та використання техніки «спуфінг» для перенаправлення користувача, який атакується, на заданий проксі-сервер, шо подовжує ланку, яку проходять пакети при UDP-з'єднанні.

При використанні техніки скрапінгу основною задачею є саме аналіз запитів та створення умов для їх подальшого відтворення При цьому відсутні елементи спуфінгу та прослуховування вебтрафіку. Скрапінг є осмисленим як певна міра взаємодії лише при наявності мережі, у якій є мінімум два пристрої, один з яких має виконувати роль вебсервера, а інший – клієнта (у класичній клієнт-серверній архітектурі, виключаючи в даному випадку технології P2P).

Через те, що скрапінг залежить від архітектури мережі, у якій він використовується, а точніше, від множини протоколів, які прийнято використовувати у даній мережі. Так, для мережі Інтернет використовується стек протоколів TCP/IP, заснований на мережевій моделі OSI, в якому можливо два види з'єднань: UDP- та TCP-з'єднання, які працюють за однойменними протоколами.

Загальну характеристику скрапінгу у мережі Інтернет можна дати наступним чином:

Види скрапінгу при загальному використанні:

- за типом використовуваних протоколів;
- за наявності або відсутності корисного навантаження в заголовках повідомлення до сервера.

Види скрапінгу у мережі Інтернет:

1) за типом використовуваних протоколів:
   - TCP;
   - UDP;
2) за типом вебзапиту:
   - GET;
   - POST;

3) за наявністю корисного навантаження (Payload):
   - є;
   - немає;
4) за наявністю параметрів в URL:
   - є;
   - немає.

1.2.2 Парсинг

Парсинг – це розбір та аналіз наданого масиву відносно структурованої інформації з метою виокремлення її піделементів. Набір даних, який підлягає процедурі парсингу, має відповідати вимогам, за якими він є структурованим та однорідним – тобто таким, який має певну внутрішню ієрархію та чия ієрархія зберігається протягом усього масиву даних.

На відміну від процесів скрапінгу та вебкраулінгу, процес парсингу може бути здійснений поза мережею та окремо від запитів до серверів, а саме – на масиві будь яких даних, що відповідають зазначеним вимогам – тобто це можуть бути текстові файли з осмисленим текстом, таблиця в реляційній базі даних або потік виводу результату роботи іншої програми.

1.2.3 Краулінг

Краулінг – це процес пошуку та індексації вебсторінок у мережі WWW. Мережа WWW або World Wide Web – це мережа текстових сторінок, які мають розширення файлу “html” та носять назву «вебсторінка», яка базується на мережі Інтернет.

Вебкраулінг або просто краулінг отримав свою назву від англійського слова “Crawl” – що у перекладі означає «Повзти». Ця назва даної технології та аналогія з павуками не є випадковістю, адже механізми вебкраулінгу мають схожі елементи з принципами життєдіяльності цих комах, а назва мережі, у якій використовується краулінг – «Веб», або англійською “Web” – перекладається, як «Павутиння».

Перед початком опису принципів роботи вебкраулінгу опишемо принципи, на яких він базується.

Сам процес краулінгу можна умовно поділити на два основні етапи:

– пошук заданої інформації у загальному масиві даних;

– процес індексації знайденого (відфільтрованого) масиву даних від загального.

Найчастіше процес краулінгу відбувається на базі даних усієї інтернет-мережі, та у подальшому будемо вважати, що мова їде про індексацію масиву інформації мережі Інтернет, хоча у своїй сутності даний процес може відбуватися й у інших масивах взаємопов’язаних графо- або деревоподібних структур даних за типом самих даних та/або атрибутами, які також залежать від типу файлу. Тобто, основна проблематика пошуку у масиві (полі) даних – це неоднорідність та найчастіше неповність даних, а також великий масив типів атрибутів, за якими треба оцінювати файли або їх зміст та зв’язувати ці показники з атрибутами.

Наведемо декілька прикладів цієї проблеми, спираючись на той факт, що усі дані загально можуть бути або людиночитанні, або ні. При розгляданні поняття «людиночитанні» варто розуміти, що це можуть бути не тільки текстові данні, але й будь-які, які може зрозуміти людина (у широкому сенсі слова «зрозуміти» – не беручи до уваги мову, якою викладено текстові дані, якщо мова йде по них) – тобто інтерпретувати та вилучити нову інформацію чи факти без використання допоміжних засобів перетворення інформації – її очищення чи зміни формату представлення.

Під наведене формулювання підпадає наступний вид даних:

– текстові данні;

– зображення.

Для інших типів даних, таких як аудіофайли та відеофайли, процес пошуку та індексації може дещо відрізнятися.

Як було зазначено, для кожного типу файлів існує свій набір параметрів для індексації. Для вищенаведених основними метриками є:

1) для текстових файлів: входження заданих слів пошукового запиту у корпус тексту. Також можуть застосовуватися техніки та алгоритми NLP (Natural Language Processing), який надає змогу частково аналізувати частини мови та сенсове навантаження за допомогою застосування направлених числових векторів, або як їх ще називають – векторів слів. Ця методика дозволяє будувати пошукові індекси для текстів на підставі значень векторі слів, тому що в основі цього принципу полягає групування векторів, що мають найбільшу схожість – тобто розташовуються поряд у загальному просторі векторів, а значить мають близьке синусове навантаження;

2) для зображень: кількість пікселів (роздільна здатність зображення), порядок пікселів (наприклад для індексації віддзеркаленої версії зображення) та їх характеристики, до яких належить: співвідношення кольорів RGB (для колірного простору RGB), контраст (рівень білого, до загального співвідношення усіх трьох кольорів пікселю). Також, можуть бути використанні метадані файлу зображення, до яких можна віднести: дату та час зйомки, можливі дані про апаратуру на яку був зроблений знімок та у дуже рідкісних випадка, які залежать більш від виробників фото- та відеотехніки, які можуть виставляти у налаштуваннях своєї продукції, геодані – де саме було зроблено фотографію чи проведено відеозйомку.

3) для аналізу та подальшої індексації відеофайлів можливо використовувати той самий підхід, що й для аналізу та індексації зображень, але для більш точного результату необхідно використовувати частину з алгоритмів, які на сьогоднішній день застосовуються у сучасних відеокодеках, наприклад AV1 або H.266 (VVC) [8], а саме – алгоритми передбачення наступного кадру.

4) індексація аудіофайлів, в свою чергу, може бути прямою чи не прямою – тобто, можна індексувати знайдені файли, наприклад, по таким чисельним характеристикам, як максимальна амплітуда звучання у певній проміжок відтворення файлу, або проводити індексацію, використовуючи вже зазначені методі NLP для отримання сенсу людської мови, що може бути записана у аудіофайлі. Але ці обчислення потребують дуже значних витрат або занадто великий інтервал часу, що не є прийнятним для кінцевого користувача.

Підходячи до задачі пошуку у контексті вебкраулінгу, варто звертати також уваго не тільки на типи шуканих файлів та атрибути для їх індексації, а також на наступні параметри:

1) розмір файлу або загального корпусу даних чи його окремої частини, за якої відбувається пошук;

2) де саме буде відбуватися пошук – у розподіленій мережі або на локальному накопичувачі, – адже від цього залежить, які саме алгоритми пошуку будуть найбільш ефективними;

3) чи потрібна у системі підтримка неповних або нечітких (нечітко сформульованих запитів);

4) чи має пошук бути виконаний на декількох мовах, якщо виконується пошук за текстовими даними.

Також, для систем пошуку важливо забезпечити підтримку системи «чорних» та «білих» списків для виключення або дозволу певних результатів з загального набору.

Розглянемо питання індексації пошукових результатів. Відповідно до визначення процесу індексації, яке було наведено вище, можна сказати, що індексувати файли можна потоково – тобто у реальному часі – або кластерно. При потоковій індексації дерево пошуку перебудовується безперервно, за індексами, що постійно змінюються. Для кластерної індексації зазвичай використовується кешування результатів та процес перекластеризації у певний момент часу або при виконанні певних умов.

Через те, що основою ідеї вебкраулінгу є автоматизований пошук та зв’язування результатів шляхом їх індексації, є певний сенс звернути увагу на такі варіанти розширення пошукових результатів:

- перевірка орфографії знайдених текстових результатів;
- пошук схожих (синонімічних) значень;
- виконання класифікації пошукового запиту перед початком пошуку та пошук у окремих групах в залежності від результатів класифікації.

Але, на відміну від звичайного пошуку, у вебкраулінгу не використовується:

1) ранжування даних. Основною метою краулінгу, на відміну від пошуку, є не надавання відсортованих за релевантністю даних для кінцевого користувача, а забезпечення як можна повного масиву даних для подальшого аналізу та можливої обробки;

2) пошук копій;

3) фільтрація «малокорисних» документів, чи результатів пошуку.

Ознайомившись з базовими принципами, на яких будується подальша робота вебкраулера, розглянемо безпосередньо принципи його роботи.

При роботі у мережі Інтернет та використанні протоколів http або https існує стандартний перелік кодів відповідей від вебсервера, які можна поділити на декілька основних груп:

1) Коди відповідей 1xx – Інформаційні коди;
2) Коди відповідей 2xx – Повідомлення про успішність виконання запиту;
3) Коди відповідей 3xx – Перенаправлення на іншу вебсторінку;
4) Коди відповідей 4xx – помилки у запити зі сторони клієнта;
5) Коди відповідей 5xx – внутрішні помилки з боку вебсервера.

На підставі наведених груп кодів відповідей можна виділити три основні стани вебкраулера: успішне виконання запиту, перенаправлення, помилка. Найбільш цікавим з цих варіантів є ситуація з перенаправленням на іншу сторінку, яка може спричинити циклічних обхід двох взаємопов’язаних сторінок. Для уникнення цієї ситуації, необхідно виконувати порівняння двох посилань на ідентичність та їх попередню нормалізацію, яка забезпечить єдиний формат та представлення посилань, перед їх порівнянням.

Таким чином, роботу найпростішого вебкраулера можна описати наступним чином:

Крок 1. Отримання сирцевого коду початкової сторінки, яка є відправним пунктом для виконання процедури краулінгу.

Крок 2. Якщо код відповіді вебсервера входить до групи сигнальних кодів 2xx або 3xx – перейти до Кроку 3, інакше – закінчити виконання.

Крок 3. Якщо код відповіді вебсервера входить до групи 2xx – запустити процес парсингу сторінки. Знайти необхідну інформацію та посилання на інші сторінки та перейти до Кроку 5. Інакше – перейти до Кроку 4.

Крок 4. Якщо код відповіді вебсервера входить до групи 3xx – повернути посилання, отримане в якості відповіді від серверу. Перейти до Кроку 5.

Крок 5. Занести посилання, які вже було відвідано – до списку відвіданих посилань. Перейти до Кроку 6.

Крок 6. Відфільтрувати список нових посилань відносно списку відвіданих посилань. Перейти до Кроку 7.

Для кожного посилання у списку посилань виконати запит до вебсервера**,** та якщо глибина пошуку менша за встановлену **–** перейти до кроку 2. Інакше – закінчити виконання.

1.3 Обґрунтування використання та приклади взаємодії наведених технологій, їх синтез задля покращення результатів пошуку інформації

Кожен елемент у ланцюгу збору та аналізу даних у мережі Інтернет та загальний аналіз даних, до якого відноситься й парсинг, залежить від іншого підходу, тобто кожен з методів обробки не є відокремленим від інших (табл. 1.2). Так, у контексті використання мережі Інтернет, парсинг вебсторінок є неможливими без їх автоматизованого отримання у тому умовному порядку, у якому вони поєднані, за що і відповідає процес вебкраулінгу.

Таблиця 1.2 – Характеристики основних технологій збору інформації

Характеристика	**Parsing**	**Scraping**	**Crawling**
Для чого використовується	Отримання певної інформації або частини інформації із завантаженого раніше джерела	Отримання інформації	Індексування вебсторінок
Особливості	Аналіз структури змісту	Звернення до API джерела або перехоплення вебзапитів	Використання «павуків»

Характеристика	**Parsing**	**Scraping**	**Crawling**
		Не використовують robots.txt	Враховують зміст файлу robots.txt
Цілі використання	Отримання інформації	Отримання інформації	Двигуни для вебпошуку
Можлива попередня стадія	Scraping	Crawling (у якості інструменту пошуку сторінки)	- Scrapping - Crawling
Відомі інструменти	Beautiful Soup Selenium	Fiddler BurpSute Консоль веббраузера для розробників	Scrapy Apache Nutch StormCrawler Selenium
Методи захисту	Часта зміна структури сайту/динамічна генерація верстки Використання HoneyPot-технік	Обмеження кількості та максимального часу запитів з одної IP-адреси Надання неповних/часткових даних Обов'язкова реєстрація для доступу до даних Використання авторизації/розмежування ролей для доступу до даних Очищення частини даних, які не потрібні у робочому процесі Використання CSRF-токенів Налаштування файлу конфігурації .htaccess – для Apache (та аналогічних файлів для інших серверів)	Використання файлу robots.txt при налаштуванні вебсервера
Методи обходу захисту	Використання об’єктів прив’язки , які базуються на контексті змісту сторінки Попередній аналіз сторінки на зразок використання HoneyPot-технік		

Характеристика	Parsing	Scraping	Crawling
Особливості використання	Будь-які відносно структуровані дані - незалежно від місця зберігання та типу сховища	Інтернет	World Wide Web

Слід зазначити, що парсинг може бути виконаний майже на будь-якому наборі даних, незалежно від його місця розташування, наприклад, на даних, які повертає база даних у відповідь на запит, або у корпусі текстових даних.

Зупинимось на особливостях та напрямках реалізації вебкраулінгу.

Можлива реалізація вебкраулінгу у двох основних технологічних напрямках:

1) вебкраулінг за допомогою повної імітації роботи вебпереглядача;

2) краулінг, який являє собою циклічний перехід за посиланням на кожній сторінці – на наступні, з певною умовою виходу.

Кожен з цих підходів має у свої основі певні переваги та недоліки. При процесі вебкраулінгу з повною імітацією роботи вебпереглядача, використовується наступний алгоритм:

Крок 1. Запуск програми-краулера;

Крок 2. Запуск спеціальної програми вебдрайвера, яка контролює основний вебпереглядач та імітує користувацькі дії в ньому, наприклад, такі як натискання клавіш чи заповнення текстових форм;

Крок 3. Вебдрайвер ініціалізує та запускає основний процес вебпереглядача.

При кожному запуску програми-краулера цей процес необхідно повторювати.

Процес краулінгу, який виконується таким чином, може зустрічати певні перешкоди та не бути досить ефективним з наступних причин:

а) мала швидкодія процесу ініціалізації та збору інформації – при кожному запуску краулера, витрачається певний час, який йде на виконання алгоритму, що описано вище;

б) мала ефективність роботи з точки зору автоматизованого процесу. Кожна дія на вебсторінці сайту, буде виконана, хоч із значно більшою швидкістю, ніж дії виконуванні людиною, але з точки зору повної автоматизації та швидкодії обчислювальних пристроїв інтервал часу, що йде на виконання цих дій, є величезним, адже, для кожної сторінки вебдрайвер має:

1) отримати команду, що саме треба зробити на сторінці – натиснути кнопку, заповнити текстове поле, перейти за посиланням, виконати пошук вебелементу на сторінку за заданими атрибутами й т. і.;

2) передати відповідну команду на виконання основному процесу вебпереглядача.

В свою чергу, вебпереглядач виконує наступні дії:

1) робить https/https/ws-запит до сервера;

2) чекає на отримання відповіді вебсервера;

3) виконує рендерінг вмісту вебсторінки для відображення кінцевому користувачеві.

Як можна побачити фінальний ланцюг дій, необхідних для виконання, є досить великим, враховуючи й той факт, що в даному випадку користувач не виконує взаємодії з веббраузером, тобто крок рендерінгу сторінки для її відображення є непотрібним. З іншого боку, такий підхід дозволяє виконувати обхід тих сторінок, що формуються динамічно з боку клієнта, за допомогою скриптів, а не зберігаються сервером у заздалегідь підготовленому вигляді.

Вебкраулінг, в якому застосовується обхід заданої множини посилань – тобто, який можна звести до задачі побудови та обходу ієрархічної структур даних, відомої як «дерево», – є більш оптимальним, але підходить лише для «статичних» вебсторінок. Також, в обох випадках, додатковими перетинами, які можуть викликати суттєві проблеми у роботі програми, є:

1) значення налаштувань з боку вебсервера, вебсторінки якого підлягають процесу вебкраулінгу. Наприклад, дуже часто багато вебсерверів мають у списку своїх налаштувань такі параметри, як максимальна кількість запитів за певний

період часу, після яких на невеликий проміжок часу користувачеві забороняється робити будь-які запити;

2) додатковими механізмами, що відстежують активність користувача є:

- вимірювання середньої тривалості дії, яку виконує користувач;
- відстежування заголовків запиту, а саме, заголовків для управління політикою переходу між вебсторінками;
- використання cookie-файлів;
- використання внутрішнього кеш-сховища браузера для зберігання токенів авторизації, за якими також можна виконувати ідентифікацію користувача;
- використання невидимих вебелементів у якості тригерів, які зазвичай може запустити користувач з необережності та через той факт, що він їх не може побачити, але автоматизована система – може працювати тільки з вказаними елементами та тим самим викликати підозру у перевіряючої системи;
- виконання перевірки наявності патернів руху курсору;
- використання системи “Capcha” для перевірки підозрілих дій, що виконуються скриптом у автоматичному режимі.

Обговоримо наведені проблемі, які притаманні для обох версій вебкраулінгу.

1.3.1 Вимірювання середньої тривалості дії, яку виконує користувач

Як вже було зазначено раніше, одною з проблем для загальної швидкодії системи, що розроблюється – є «зайві» кроки виконання при процесі краулінгу, що значно знижують швидкодію при великих обсягах масиву вебсторінок для обходу, але з точки зору іншої, аналізуючої системи, швидкість виконання дій автоматизованої системи все одно є значно більшою за звичайне середньостатистичне виконання тієї самої операції – людиною.

1.3.2 Відстежування заголовків запиту

Певні види заголовків вебзапиту використовуються для побудови певного маршруту, за яким можна зрозуміти, з якого іншого ресурсу користувача було перенаправлено на даний. Зазвичай такі заголовки використовуються разом з відповіддю яка має код відповіді з групи 3xx – тобто тих, що відповідають за перенаправлення користувача. До типу заголовків, про які йде мова, належать:

1) Referer – URI ресурсу, з якого було здійснено перенаправлення;
2) Location – URI, за яким клієнтський пристрій має перейти, або URI спеціально створеного ресурсу.

Унікальна комбінація значень цих URI-заголовків дає можливість частково ідентифікувати користувача.

1.3.3 Використання cookie-файлів

Cookie-файл – це спеціалізований текстовий файл, який зберігає певну інформацію про користувача, яка дозволяє виконувати його ідентифікувати. Самі cookie-файли не можуть та не повинні зберігати ніякої чутливої інформації, наприклад, такої як логіни, паролі або хеш-значення паролів. Однак, cookie-файли є зручним та відносно безпечним інструментом, який дозволяє: ідентифікувати користувача, або групу осіб, на підставі певних значень самих cookie-файлів, зберігати інформацію. Про користувацьку сесію та налаштування сторінки для їх подальшого відновлення, наприклад, після перезапуску веббраузера. Будова файлу являє собою певну кількість пар у форматі «атрибут=значення;» (кожна пара атрибут – значення повинна відділятися символом «крапка з комою»). Відповідь від вебсервера для встановлення cookie-файлу має наступну структуру:

1) Set-cookie: ім’я атрибуту = значення;
2) Max-age = значення у секундах;
3) Comment = текстовий коментар;
4) Domain = ім’я доменної зони, до якої належить сервер;

5) Secure = значення політики безпеки, яка встановлює, чи повинен веббраузер клієнтського пристрою передавати поточний cookie-файл назад до вебсервера при повторному відвідуванні сторінки через захищене з'єднання;

6) Version=1 – версія файлу.

Варто відзначити, що обов'язковими для cookie-файлів є лише пара атрибута та його значення (чи декількох атрибутів та їх значень) параметр – Version. Унікальна комбінація атрибутів та/або їх значень – також надає зловмиснику чи ворожій системі – ідентифікувати користувача.

Висновки до розділу 1

Як можна побачити, розробка програмного забезпечення, яке б поєднувало усі наведені класи алгоритмів для виконання автоматичної модерації – є занадто дорогим та складним. Такий клас ПЗ можна було б віднести до типу надсистем – тобто таких систем, створення яких є неможливим, або занадто довгим. Саме тому на даному етапі розвитку комп'ютерних наук та суміжних областей для досягнення точного результату є найбільш підходящім використання модерації, що здійснюється людиною або групою людей та може включати в себе експертну базу або окрему системи підтримки прийняття рішень. Однак, для виконання даного виду модерації перш за все необхідно першочергово зібрати певний масив інформації, після чого виконати його аналіз, очистку та структуризацію, що саме й полягає у основі роботи системи, яка має бути розроблена при виконанні даної кваліфікаційної роботи.

2 ВИБІР КОДОВОЇ БАЗИ ТА НАБОРУ БІБЛІОТЕК ДЛЯ РОЗРОБКИ ПРОЄКТУ

2.1 Вибір мови програмування та аналіз сучасного стану її екосистеми

Для розробки зазначеного проєкту було проаналізовано та обрано декілька початкових варіантів мов програмування, кожна з яких мали певні переваги та недоліки. Основний список обраних мов програмування – складався з наступних пунктів:

- C++;
- C#;
- Python.

Коротко розглянемо особливості кожної з мов.

2.1.1 Мова програмування C++

Дана мова програмування відноситься до класу строго типізованих статичних компільованих мов програмування; Це означає, що основні типи даних усередині мови є чітко визначеними та мають бути відомі заздалегідь на момент компіляції програми. Кожен з типів даних має строго визначену ієрархію, за якої він може бути перетворений явним чином на інший, дозволений тип, за що й відповідає строга типізація.

Ця мова програмування належить до класу компільованих мов – тобто, сирцевий код програми проходить декілька етапів, після яких створюється виконуваний файл, здатний до самостійного запуску. Особливістю цієї мови програмування є можливість та необхідність роботи з постійною оперативною пам’яттю пристрою – тобто на відносно низькому рівні – за допомогою таких структур даних, як вказівники. Ця особливість може спричинити як значне прискорення роботи програми через оптимізацію роботи з пам’яттю, так і спричинити так званий «витік пам’яті» при некоректному її використанні.

2.1.2 Мова програмування C#

Мова програмування C#, як і мова C++, також є компільованою, строго типізованою, статичною мовою, але на відміну від C++ має автоматизоване управління виділенням та очищенням робочої пам’яті за допомогою технології та однойменного класу алгоритмів – “Garbage Collector” (або просто GC), – яка дозволяє виділяти ділянку пам’яті для змінної, відстежувати кількість посилань на цю змінну у сирцевому коді та під час виконання програми та за необхідністю видаляти виділене місце, якщо змінна чи об’єкт більш не використовується.

Основною сильною стороною та основним недоліком цієї мови програмування є екосистема платформи “.NET”, яка забезпечує легке перенесення створеної програми на частину інших платформ та операційних систем за допомогою використання проміжного коду CIL (Common Intermidiate Language), який може у подальшому бути виконаний за допомогою системи CLR (Common Language Runtime), яка входить до складу платформи .NET Framewok. Головною проблемою даної платформи є відносна нерозповсюдженість, на відміну від її найближчого конкурента – мови програмування Java та її технології програмної віртуалізації Java Virtual Machine (JVM).

2.1.3 Мова програмування Python

На відміну від мов програмування C++ та C#, мова Python є інтерпретованою мовою програмування, а не компільованою Це означає, що для її виконання не є обов’язковим створення вихідного виконуваного файлу – обробленого сирцевого коду компілятором – та побудова лексичного та семантичного дерева.

Інтерпретовані мови програмування виконуються прямо під час зчитування їх сирцевого коду інтерпретатором та не підлягають додатковій зміні чи обробці самим інтерпретатором, якщо тільки це не було задумано розробником. Також мова програмування Python є динамічною та не строго типізованою – тобто тип однієї й тієї самої змінної може бути змінений у процесі виконання програми, а значення

типів даних кожної зі змінних не повинні бути визначенні на момент запуску програми на виконання.

Для розробки системи перешкоджання flood- та flame-атакам було обрано мову програмування Python, зі списку доступних кандидатів, через наступні переваги:

1) відносно проста структура мови та можливість її швидкого опанування у порівнянні з C++;

2) автоматичне управління пам'яттю, але на відміну від технології GC, у C# код не є «прив'язаним» до певної платформи чи кодової бази;

3) належність до класу інтерпретованих мов, що надає змогу виконання безпосередньо, при використанні файлу скрипту з сирцевим кодом програми, без створення додаткового вихідного файлу;

4) наявність великої спільноти розробників;

5) платформонезалежність;

6) великий вибір готових модулей-бібліотек, зручних та простих для застосування, які можна легко встановити за допомогою одного з багатьох – так званих «пакетних менеджерів» – спеціальної програми, яка дозволяє легко керувати сукупністю встановлених бібліотек та встановлювати нові;

7) можливість вирішення широкого класу різнорідних завдань – від створення, редагування та запису локальних файлів до роботи з великою кількістю інтернет-протоколів та обробки великих даних на кластері розподілених систем.

Відносна простота мови програмування Python також робить її найбільш вдалим вибором для швидкого виконання прототипування програмного забезпечення та легкого вчинення змін.

Варто зазначити, що існує дві основні версії даної мови програмування – версія 2.x та 3.x, – які варто розрізняти через суттєві зміни у самій архітектурі мови та багатьох інших суттєвих змін, що вплинули на її подальший розвиток.

Ще однією перевагою Python є наявність декількох гілок реалізації ядра мову, а саме:

1) Cpython – першочергова реалізація мови, заснована на C++;

2) IronPython – реалізація мови, яка використовує мову програмування c# та її платформу .NET (з недавнього часу, ще й кросплатформну реалізацію .NET – бібліотеку Mono);

3) PyPy – реалізація мови Python, виконана за допомогою мови програмування Python, та інші.

Кожна з наведених реалізацій відрізняється відносною швидкодією та певним колом задач, для яких вона була створена.

2.2 Використання віртуальних оточень

Обрана мова програмування працює у сукупності з інтерпретатором – спеціальною програмою, яка рядок за рядком виконує написаний сирцевий код.

Також, у середовище інтерпретатора, тобто у ту директорію, де він знаходиться, додаються необхідні у процесі роботи бібліотеки з готовою реалізацію певного функціоналу. При роботі з декількома проєктами паралельно або при великій кількості бібліотек, які були встановлені для попередніх проєктів – виникає ситуація, яка називається «проблема залежностей» – стан, при я кому різні версії однієї й тієї самої бібліотеки або декілька окремих бібліотек – можуть заважати один одному, порушуючи роботу усього проєкту, наприклад, через спільне використання системних бібліотек, але для кожної бібліотеки мови Python – буде необхідна системна бібліотека певної версії. Також, додаткової плутанини та шкоди інтерпретатору, який встановлено у операційній системі за замовчуванням, може спричинити наявність не тільки бібліотек, а й інтерпретатору ще однієї версії, бібліотеки якого можуть почати конфліктувати з бібліотеками основної версії інтерпретатору.

Для уникнення цієї проблеми розробниками мови Python була створена технологія віртуальних оточень.

Віртуальне оточення (англ. Virtual Environment, надалі VE) – це окрема директорія, як містить у собі копію виконуваного файлу інтерпретатору, пакетного менеджеру та усі залежності – тобто файли усіх зовнішніх, додаткових бібліотек, які потрібні для коректної роботи проєкту. Принцип роботи VE полягає у тому, що

для кожного проєкту створюється своє оточення, у якому використовується своя копія інтерпретатора, незалежна від системних (глобальних, встановлених у межах усієї системи для такого ж глобального інтерпретатору) бібліотек та яка «бачить» та працює лише з залежностями та бібліотеками, які знаходяться усередині директорії віртуального оточення, а інсталювання нових бібліотек – також виконується у середину віртуального оточення. Також, VE – вирішує й проблему з використанням різних версій інтерпретатора на одному пристрої.

Продемонструємо приклад встановлення системи віртуальних оточень virtualenv на операційній системі Linux та створення нового віртуального оточення.

Для встановлення системи віртуальних оточень необхідно виконати команду:

sudo apt-get install python3-virtualenv

Для створення нового VE:

virtualenv –python=python<ver><env_name>,

де <*ver*> – чисельна версія існуючого інтерпретатора Python, наприклад 3.9;

<*env_name*> – довільне дозволене ім’я нового віртуального оточення (без спеціальних та інших заборонених для найменування символів).

Кожне VE має таке поняття, як “активація” – тобто процес, при якому усі подальші команди користувацького командного терміналу, які стосуються мови програмування, інтерпретатору або його оточення, будуть виконанні по відношенню до інтерпретатора у середині “активованого” віртуального оточення.

Для активації віртуального оточення необхідно виконати команду:

source <env_name>/bin/activate,

де <*env_name*> – шлях до директорії віртуального оточення [9].

При створенні віртуального оточення у його директорію копіюється виконуваний файл інтерпретатора, а також створюються посилання на системні файли, від яких залежить інтерпретатор. При активації VE система віртуальних

оточень тимчасово заміняє шлях до інтерпретатора Python у змінній системного оточення *Path*, що надає операційній системі можливість виконувати пошук інтерпретатора у потрібній директорії.

Безпосередньо сам інтерпретатор, при своєму запуску виконує рекурсивний пошук у поточній директорії та у директорії на рівень вище – файлу *os.py*. Саме за наявністю цього файлу інтерпретатор й розуміє, яка саме директорія є директорією модулей – тобто тою, що містить у собі потрібні зовнішні та додаткові бібліотеки.

Суттєвою перевагою Python є можливість роботи з програмою як у вигляді скрипта, так і окремого виконуваного файлу, у який додано копію інтерпретатору, що робить його повністю незалежним від оточення середовища операційної системи, у якій буде виконуватись запуск.

2.3 Обґрунтування вибору бібліотек для розробки проєкту

Для забезпечення виконання процесу парсингу вебсторінок було обрано бібліотеку “Beautiful Soup”, яка надає можливість проводити лексичний аналіз HTML- та XML-дерев, розбираючи їх структуру та виконуючи представлення її окремих елементів у форматі спеціальних об’єктів – вузлів синтаксичного лексичного дерева, для кожного з яких є можливість виконувати переміщення відносно даного вузла до інших сусідніх, які знаходяться на одному з ним рівні, до батьківських або до вузлів-нащадків. Наведемо кілька прикладів можливостей даної бібліотеки:

- пошук тегів елементів вебсторінки за їх назвою;
- пошук тегів елементів вебсторінки за текстом, який міститься усередині тегу;
- пошук тегів елементів вебсторінки за їх назвою та фільтрація за певним значенням атрибутів знайденої підмножини тегів;
- отримання піделементів певних тегів;
- виконання пошуку за паттернами регулярних виразів;
- отримання коментарів до вебтегів, які були залишені розробниками;
- обхід сторінки вшир та вглиб;
- отримання списку елементів, що підпадають під певний критерій;
- виконання пошуку, спираючись на CSS-елементи сторінки.

Також, бібліотека надає можливість не тільки виконувати пошук елементів, а й їх зміну чи часткове видалення нащадків елементів чи їх самих, тим самим отримуючи модифіковану версію вебсторінки. Для більш зручного використання сирцевого коду вебсторінки людиною, також має місце функціонал з форматування коду усієї сторінки чи її елементів.

2.3 Використання бібліотеки Selenium для автоматизації взаємодії з веббраузером

Selenium – це спеціалізоване програмне забезпечення з відкритим сирцевим кодом, яке розповсюджується під ліцензією Apache License 2.0. Основними функціональними особливостями цього продукту є можливість автоматизації дії різних версії вебпереглядачів, таких як: Googlr Chrome, Opera, Firefox, що робить його ідеальним інструментом для автоматизації тестування відображення та поведінки веб – сайтів у різних умовах. Але, цей інструмент також можна використовувати й для задач вебкраулінгу, адже він не просто імітує роботу повноцінного вебпереглядача, а надає спеціалізований програмний інтерфейс, для керування його роботою, що дозволяє використовувати найновіші версії веббраузерів з усіма нововведеннями та гарантувати коректну роботу алгоритмів вебкраулінгу та парсингу – шляхом повного та коректного завантаження і відображення вебсторінки.

Сам проєкт Selenium, точніше, його окрема частина Selenium Web Driver, не має монолітної структури та складається з певних частин (рис. 2.1).

Умовно роботу цього ПЗ можна поділити на дві частини: бібліотеки підтримки різних мов програмування та кодом для роботи з спеціалізованим об’єктом вебдрайверу, між якими існує універсальний програмний інтерфейс взаємодії, який не залежить ні від мови програмування з одного боку, ні типу об’єкту вебдрайверу, що використовується для різних версій веббраузеру – з іншого боку.

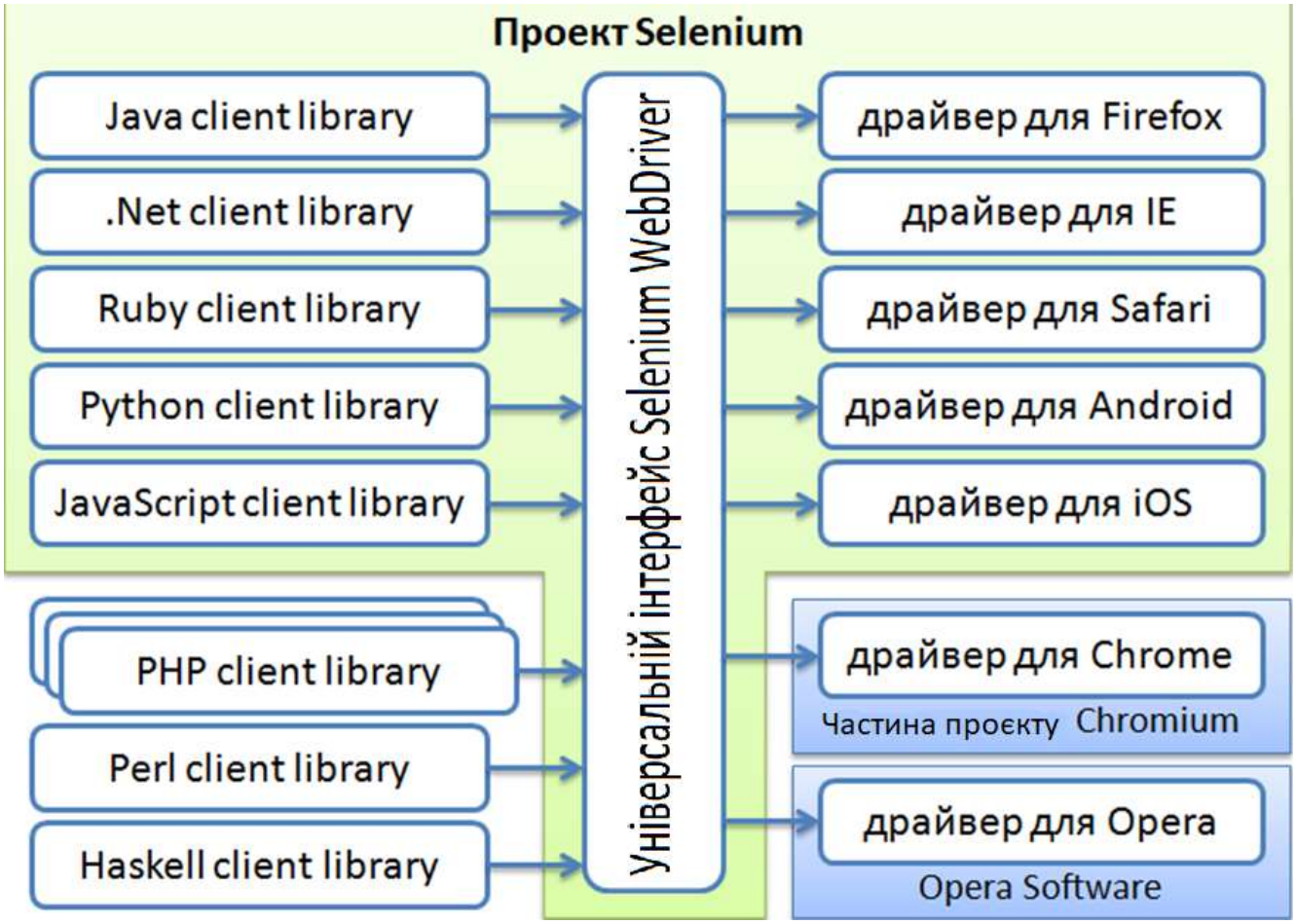


Рисунок 2.1 – Складові частини проєкту Selenium

Також, у рамках розвитку проєкту Selenium існує ще декілька продуктів – окрім Selenium Web Driver, – кожен з яких має певну спеціалізовану функцію, а саме:

1) Selenium RC – програмне забезпечення, з основ якого взяв свій розвиток Selenium Web Driver. На даний момент не знаходиться у стадії активної розробки, через значні часові та грошові витрати, які потребує для своєї підтримки. Є частково законсервованим, призупинена активна стадії виправлення вразливостей та багів;

2) Selenium Server – серверне програмне забезпечення, яке має змогу керувати вебпереглядачем не тільки на тому самому пристрої, де знаходиться сценарій автоматизації, вебдрайвер та сам вебпереглядач, а з використанням мережевого зв'язку виконувати автоматизовані дії певного сценарію, пересилаючи команди вебдрайверу, який разом з веббраузером знаходиться на іншому пристрої;

3) Selenium Grid – ще одно мережеве відгалуження продукту, яке дозволяє не тільки запускати дії на віддалених хост-пристроях, як це робить Selenium Server, а робити це масово, використовуючи мережеву топологію «зірка, в центрі якої знаходиться виділений сервер, який надсилає певні, окремі команди кожному з віддалених вузлів, на яких запущена певна версія веббраузера, що дозволяє не тільки прискорити процес автоматизації тестуванні, протестувавши один й той самий вебсайт на великій кількості вебпереглядачів, а й значно прискорити процес вебкраулінгу, розподілено запустивши певну частку обходу дерева сторінок в окремих версіях веббраузерів. Такий підхід до вебкраулінгу не тільки має певні переваги з точки зору зручності централізованого керування, а ще й значно зменшує ризик виявлення автоматичними системами анти-краулінгу, бо сам процес ведеться у декількох частинах – кожна на своїй версії вебпереглядача – що й імітує поведінку декількох звичайних користувачів;

4) Selenium IDE – застосунок-розширення для веббраузеру Firefox, яке дозволяє “записувати” дії, які виконує користувач та або відтворювати їх прямо у веббраузері, або робити експорт у один з форматів, що підтримується розширенням, серед яких є і варіант експорту в автозгенерований сирцевий код на мові Python.

2.4 Огляд альтернативних засобів виконання процесу збору інформації з вебсторінок

Технічні засоби та методи для виконання краулінгу та парсингу, які було розглянуто, – зокрема проєкт Selenium – не є єдиними засобами, які можна використовувати для досягнення поставленої мети, а в окремих випадках ці інструменти є не досить зручними або підходящими. Для уникнення певних проблем, які можуть виникати при використанні автоматизованого веббраузеру, такі як: необхідність підтримки актуальної версії вебдрайверу для керування веббраузером, наявність систем для розпізнавання автоматизованих дій та необхідність встановлення додаткових компонентів – альтернативним варіантом є застосування, скриптових мов програмування, використання яких можливо у

інтегрованому середовищі звичайного веббраузеру, наприклад, у консолі розробника, яка за стандартом доступна для користувачів при натисканні спеціалізованої клавіши “F12”. Розглянемо основні варіанти скриптових мов, які можуть бути використанні для виконання парсингу та краулінгу.

JavaScript. JavaScript – мова програмування яка бере свій початок, як діалект мови Scheme, яка в свою чергу є діалектом мови програмування Lisp. Сама мова була розроблена у 1995 році компанією Netscape Navigator, яка розробляла однойменний веббраузер. З плином часу, до процесу розробки доєдналася компанія Microsoft, яка вирішила розробляти окрему версію даної мови, внаслідок чого виникла суперечка та часткова несумісність двох версії від різних компаній. Виходом з цієї проблеми стало створення стандарту ECMAScript – Європейською асоціацією виробників комп’ютерів (European Computer Manufacturing Association). Саме процес стандартизації даної мови дозволив досягти її широкого розповсюдження. На сьогоднішній день мова JavaScript може використовуватись не тільки у середовищі веббраузерів, алей й у якості основної мови деяких контейнерів, наприклад, таких як технологія Electron, яка надає можливість виконувати скриптові сценарії на пристроях та у середовищах, які не були до цього пристосовані, наприклад у нативних Android-застосунках. Сама мова відноситься, як і Python, до класу інтерпретуємих мов з динамічною нестрогою типізацію. Основною перевагою для використання даної мови у рамках виконання вебпарсингу та краулінгу є її безпрецедентна розповсюдженість – не тільки у рамках вебзастосунків на персональних комп’ютерах, а також й у ряді прошивок деяких мікроконтролерів, де вона використовується для реалізації певної логіки.

Ця особливість дозволяє гарантувати легкий процес та швидкість розгортання готової до роботи системи на користувацьких пристроях без встановлення додаткових залежностей та бібліотек. Але, при цьому має місце і ряд недоліків:

- відсутність строгої типізації;
- високий поріг входження;
- наявність великої кількості залежностей.

Розглянемо дані мінуси мови програмування більш детально.

Відсутність строгої типізації. Дана особливість будови мови може призводити до виникнення певних неточностей та “багів”, причиною для яких є логічні помилки програмістів, які можуть писати некоректний з точки зору типізації код, але при цьому програма може працювати коректно до певного моменту. Це може траплятись через людський фактор та можливість спільного використання двох типів даних, які насправді не є сумісними.

PHP. PHP – скриптова мова програмування серверного типу, тобто така, яка зазвичай використовується з боку сервера, або, як її ще називають backend-частина сайту. Дана мова програмування має широкі можливості для аналізу вузлів XML- та HTML-документів завдяки використанню широкого спектру бібліотек, наприклад, таких як [20]:

– Guzzle;
– Goutte.

Обидві бібліотеки є чудовим зразком швидких та надійних бібліотек для парсингу вебсторінок.

Висновки до розділу 2

У даному розділі було проведено порівняльний аналіз найбільш популярних мов програмування для вирішення зазначеного кола задач. В результаті виявлено, шо мова програмування Python є найбільш підходящім інструментом для проєктування, прототипування та розробки системи збору та аналізу інформації при використанні технологій вебзапитів та для автоматизації процесу пошуку, індексації, вилучення та каталогізації отриманої інформації.

Було описано бібліотеки сирцевого коду, які було використано у рамках розробки проєкту – їх основні переваги та властивості. Зокрема, особливу увагу було приділено бібліотеці Selenium та реалізації її внутрішньої модульної структури.

Також, було розглянуто альтернативні мови програмування, на яких можна вирішити поставлену задачу, але які не ввійшли до основного порівняльного списку через певні особливості реалізації чи використання.

3 ОСОБЛИВОСТІ ТЕХНІЧНОЇ РЕАЛІЗАЦІЇ ПРОЄКТУ

3.1 Опис будови структури проєкту

Загальний алгоритм роботи системи наведено на рис. 3.1.

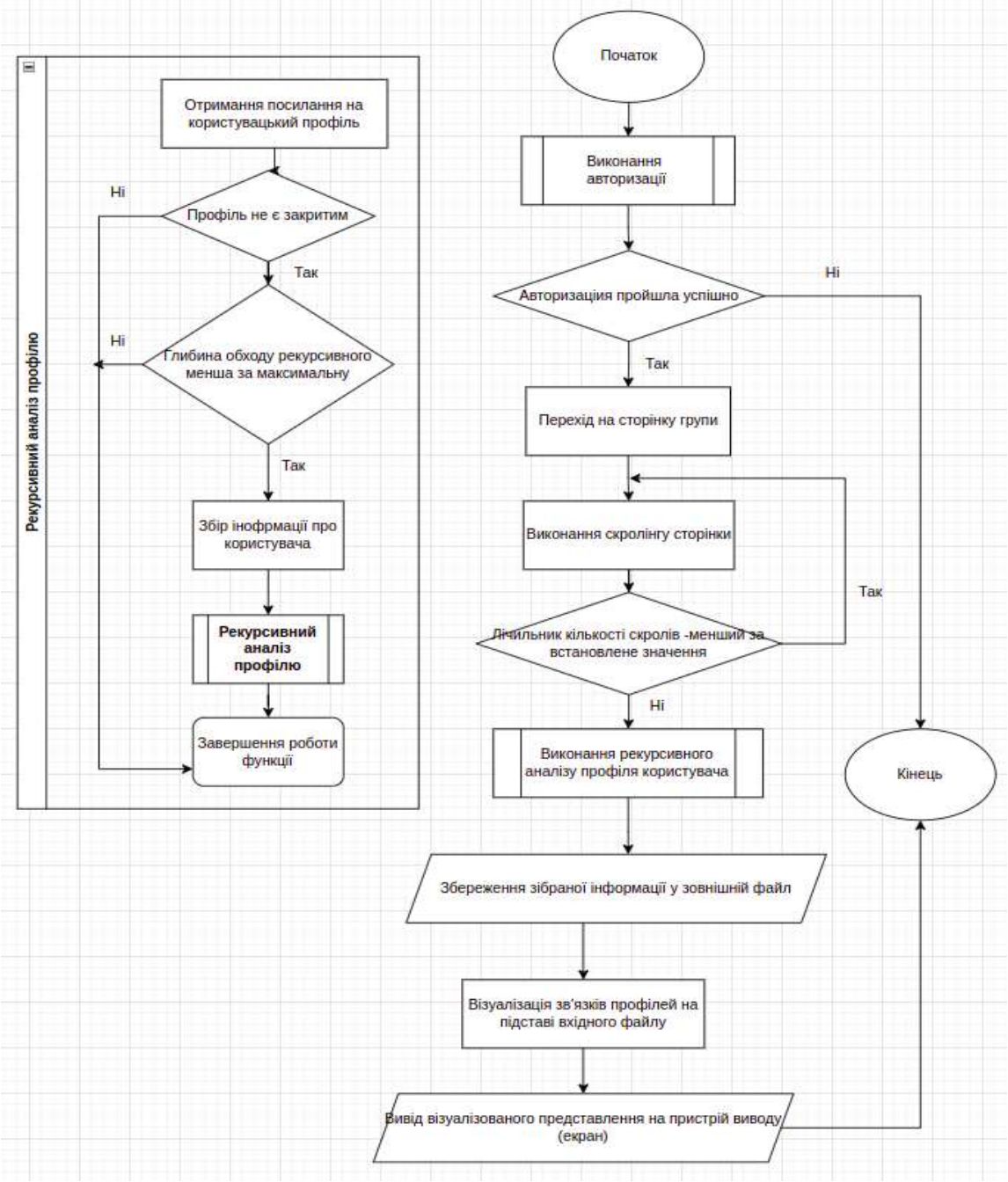


Рисунок 3.1 – Загальний алгоритм роботи системи

Як можна побачити на рис. 3.1, загальний алгоритм збору відкритої інформації для подальшого аналізу та прийняття рішень для перешкоджання flood&flame-атакам є досить зрозумілим та легким для читання та розбору людиною.

Розглянемо основні етапи роботи алгоритму, але сперш, дамо короткі визначення основним його частинам:

– виконання авторизації на сторінці входу до профілю користувача соціальної мережі Facebook з використанням наведених у конфігураційному файлі даних для даного процесу – логіну користувача та паролю.

– отримання списку посилань на профілі користувачів, які коментували певні публікації у обраних групах, за допомогою автоматизованого гортання стрічки публікацій з метою накопичення публікацій та коментарів до них, що динамічно завантажуються у цьому процесі.

– аналіз кожного профіля з отриманого списку, а саме отримання інформації з сторінки профілю та пошук взаємозв’язків з іншими, вже проаналізованими профілями. Знайденні взаємозв’язки у подальшому будуть використовуватись для графічного представлення фінальних результатів, через – побудову графу пов’язаних профілів.

Розглянемо роботу системи та особливості її реалізації більш детально.

Для покращення процесу орієнтації у файлах проєкту на рис. 3.2 наведено ієрархічну структуру його файлів та каталогів, представлену у вигляді дерева залежностей. Розглянемо структуру проєкту та його основні файли з сирцевим кодом та допоміжними файлами конфігурації.

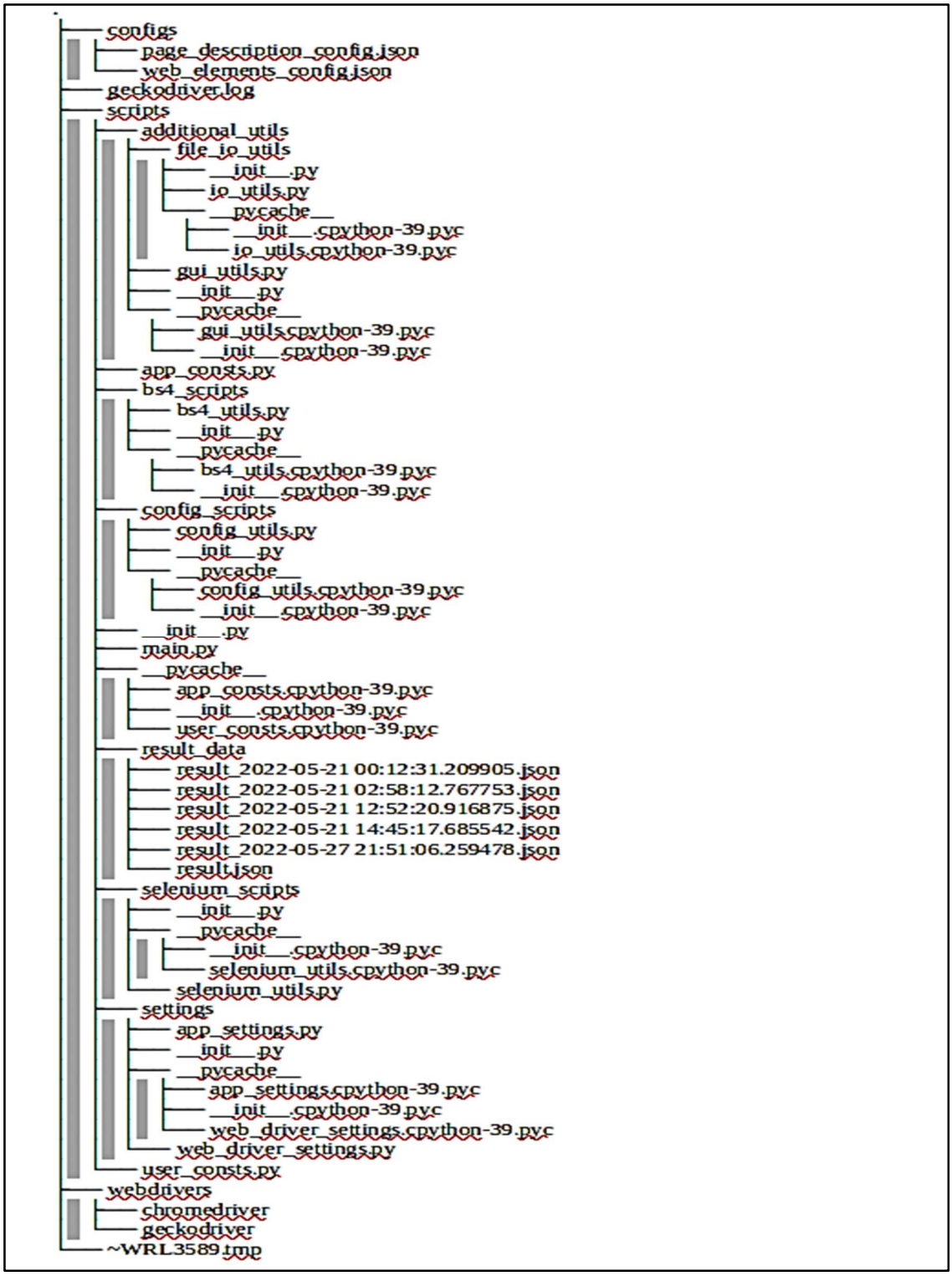


Рисунок 3.2 – Дерево файлів та директорій проєкту

Коротко опишемо структуру файлів і директорій та їх призначення.

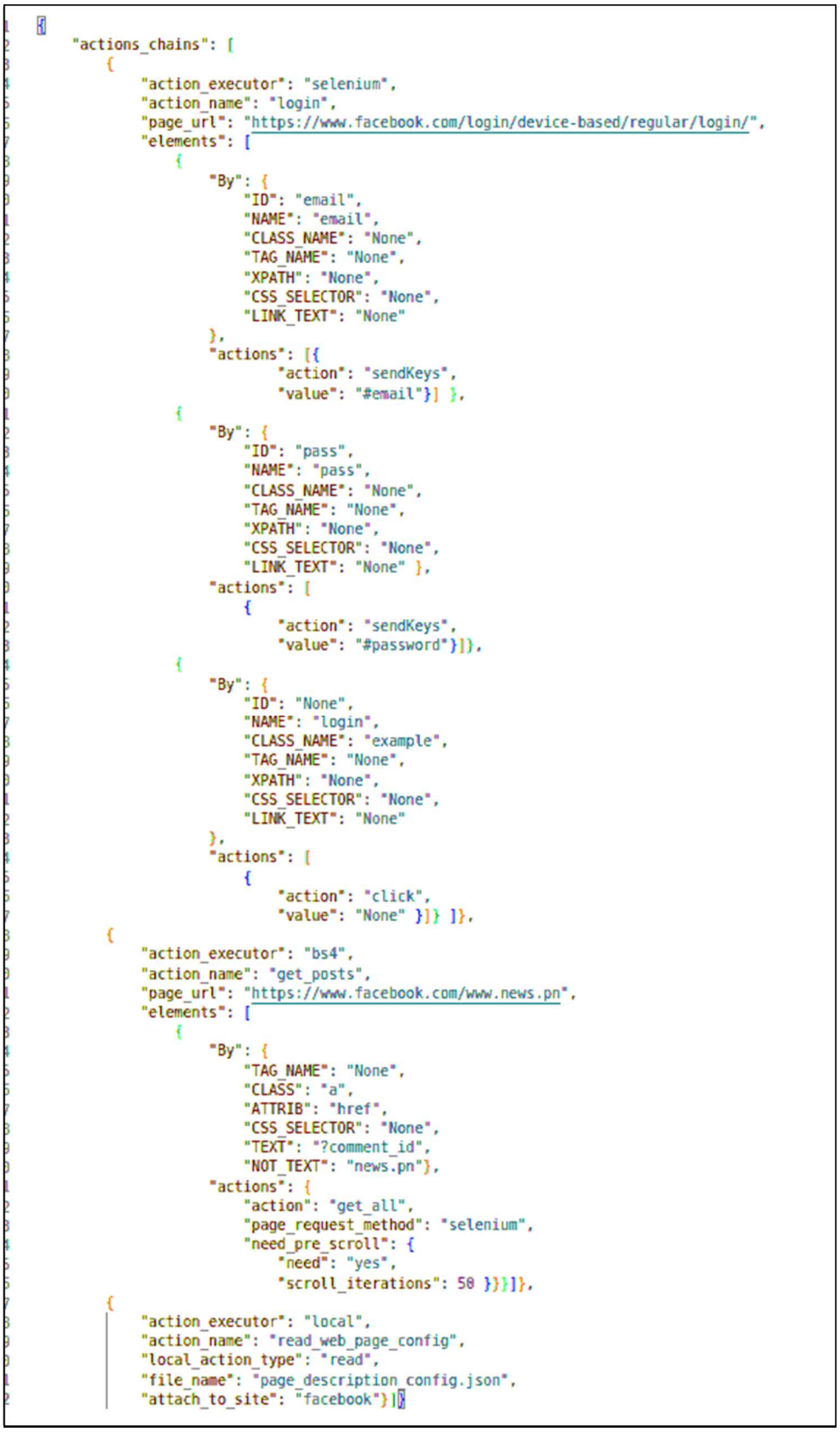
Директорія configs – містить конфігураційні файли у форматі JSON, які використовуються для налаштування роботи системи, а саме – створення низки правил за якими необхідно проводити збір інформації з вебсторінки.

Дана директорія містить два файли – два основні типи конфігураційних файлів проєкту:

а) *page_description_config.json* – файл, який відповідає за правила для сторінок, з яких необхідно отримати інформацію. Містить перелік ключових слів, за якими необхідно порозводити пошук на сторінці. Приклад шаблону для даного файлу конфігурації наведений на рис. 3.3;

Рисунок 3.3 – Шаблон файлу конфігурації

б) *web_elements_config.json* – основний *конфігураційний* файл системи, в якому зазначені дії та способи їх виконання системою. Приклад шаблону для даного файлу конфігурації наведений на рис. 3.4.



```
{
    "actions_chains": [
        {
            "action_executor": "selenium",
            "action_name": "login",
            "page_url": "https://www.facebook.com/login/device-based/regular/login/",
            "elements": [
                {
                    "By": {
                        "ID": "email",
                        "NAME": "email",
                        "CLASS_NAME": "None",
                        "TAG_NAME": "None",
                        "XPATH": "None",
                        "CSS_SELECTOR": "None",
                        "LINK_TEXT": "None"
                    },
                    "actions": [{
                            "action": "sendKeys",
                            "value": "#email"}] },
                {
                    "By": {
                        "ID": "pass",
                        "NAME": "pass",
                        "CLASS_NAME": "None",
                        "TAG_NAME": "None",
                        "XPATH": "None",
                        "CSS_SELECTOR": "None",
                        "LINK_TEXT": "None" },
                    "actions": [
                        {
                            "action": "sendKeys",
                            "value": "#password"}]},
                {
                    "By": {
                        "ID": "None",
                        "NAME": "login",
                        "CLASS_NAME": "example",
                        "TAG_NAME": "None",
                        "XPATH": "None",
                        "CSS_SELECTOR": "None",
                        "LINK_TEXT": "None"
                    },
                    "actions": [
                        {
                            "action": "click",
                            "value": "None" }]} ]},
        {
            "action_executor": "bs4",
            "action_name": "get_posts",
            "page_url": "https://www.facebook.com/www.news.pn",
            "elements": [
                {
                    "By": {
                        "TAG_NAME": "None",
                        "CLASS": "a",
                        "ATTRIB": "href",
                        "CSS_SELECTOR": "None",
                        "TEXT": "?comment_id",
                        "NOT_TEXT": "news.pn"},
                    "actions": {
                        "action": "get_all",
                        "page_request_method": "selenium",
                        "need_pre_scroll": {
                            "need": "yes",
                            "scroll_iterations": 50 }}}]},
        {
            "action_executor": "local",
            "action_name": "read_web_page_config",
            "local_action_type": "read",
            "file_name": "page_description_config.json",
            "attach_to_site": "facebook"}]}
```

Рисунок 3.4 – Шаблон загального файлу конфігурації

Директорія scripts – основна директорія для сирцевого коду проєкту, яка містить наступні дочірні директорії:

а) ***additional_utils*** **–** тут розміщуються допоміжні скрипти, наприклад, які відповідають за роботу з файловою системою **–** читання та запис оновлених конфігураційних файлів та графічним інтерфейсом **–** для відображення графу зв'язків профілів між собою;

б) ***bs4_scripts*** **–** скрипти та логіка роботи тієї частки системи, яка використовує бібліотеку *Beautifulsoup* для виконання парсингу вебсторінок;

в) ***config_scripts*** **–** містить скрипти для керування та процесу передобробки шаблонізованих конфігураційних файлів перед їх використанням.

г) ***result_data*** **–** директорія, що містить вихідні файли фінального результату роботи системи;

д) ***selenium_scripts*** **–** скрипти та логіка роботи тієї частки системи, яка використовує бібліотеку Selenium для автоматизованих дій у середовищі веббраузера та процесу краулінгу;

е) ***settings*** **–** логіка роботи, що застосовується для ініціалізації роботи вебдрайверу на етапі роботи з бібліотекою Selenium. Містить загальні налаштування запуску та управління вебдрайвером, наприклад, такі як режим перегляду вебсторінок у веббраузері – звичайний або у приватному режимі, без збереження cookie-файлів.

Також, окрім перерахованих директорій, у проєкті має місце директорія віртуального середовища, принцип роботи якого було описано раніше, каталог який містить бінарні файли вебдрайверу для веббраузера Google Chrome – версії 100 або більш ранньої та наступні файли скриптів, які містяться у корені директорії *scripts*:

а) ***app_consts.py*** **–** файл, який містить константні значення, що використовуються у оточенні системи при її роботі, наприклад**,** символ-шаблон, зазначений у шаблонному варіанті конфігураційного файлу, який буде замінено на конкретне значення, після зчитування шаблону конфігурації та перед його безпосереднім використанням;

б) ***main.py*** **–** основний файл системи, який є точкою входу до програми;

в) ***user_consts.py*** **–** містить чутливу інформацію, таку як логін та пароль від користувацького профілю соціальної мережі, які використовуються при процедурі авторизації. Зазначенні сутності винесено в окремий файл, через небезпеку їх безпосереднього використання у файлах конфігурації системи. При старті роботи ці значення будуть вилучені з даного файлу та підставлені до файлу конфігурації задля виконання дії авторизації.

Розглянувши структуру проєкту, перейдемо до детального опису алгоритму його роботи. Почнемо з файлу, який є точкою входу до виконання програми – ***main.py.***

3.2 Особливості програмної реалізації алгоритму системи

Основною ідеєю побудови системи є її відносно легка модернізації та масштабованість, завдяки системі так званих “Дій” – окремих, незалежних один-від-одного операцій, які, проте, можуть бути поєднанні у “ланцюги” та виконуватись послідовно – кожен на певному етапі роботи. Опис кожної з виконуваних дій міститься у конфігураційному файлі, що надає можливість змінювати структури та послідовність обходу вебсторінок, збору інформації та її аналізу, не змінюючи при цьому основну логіку та сирцевий код програми. Тобто, логіка обробки та алгоритмізації роботи – відокремлена від представлення основних координаційних сутностей, які і використовуються у роботі алгоритму. Кожна дія, представлена та описана у конфігураційному файлі, має певне унікальне ім’я, що дозволяє звернутися до неї та запустити на виконання за допомогою спеціалізованої функції, наприклад, як наведено на рис. 3.5.

```
if __name__ == "__main__":
    web_config_obj = read_web_element_config(config_path=path_to_json_config)
    if web_config_obj:
        execute_config_action("login")
        profiles_links = get_profiles_links(execute_config_action("get_posts"))
        web_page_config = execute_config_action("read_web_page_config")
```

Рисунок 3.5 – Точка входу в програму

Як можна побачити у наведеному вище фрагменті сирцевого коду, функція *execute_config_action*, що викликається тричі, виконує послідовно три наступні дії: *"login","get_posts"* та *" read_web_page_config".*

Розглянемо механізм виконання зазначеної дії через звернення до неї за ім'ям (рис. 3.6).

```
def execute_config_action(action_name: str):
    actions_chains_action = get_config_action(action_name)
    if actions_chains_action["action_executor"] == "selenium":
        execute_selenium_action(action_name, actions_chains_action)
        return None
    elif actions_chains_action["action_executor"] == "bs4":
        return execute_bs4_action(action_name, actions_chains_action)
    elif actions_chains_action["action_executor"] == "local":
        if actions_chains_action["local_action_type"] == "read":
            return {actions_chains_action["attach_to_site"]: read_json_file(
                path_to_json_web_page_config + actions_chains_action["file_name"])}
```

Рисунок 3.6 – Вибір гілки, яка буде виконуватись

Функція *execute_config_action* використовується для визначення типу виконуваної дії та вибору інструменту за допомогою якого цю дію необхідно виконати. Так, усі дії можна поділити на два основні типи: локальні та не локальні.

Під локальними діями мається на увазі дії, які виконуються безпосередньо з іншими файлами системи або оточення, без роботи з елементами соціальної мережі – її вебсторінками чи інформацією з неї. До локальних дій можна, наприклад, віднести взаємодію з іншим конфігураційним файлом – тобто, ситуацію, коли процес читання іншого конфігураційного файлу прописаний у ланцюгу дій основного конфігураційного файлу (рис. 3.7).

```
    {
      "action_executor": "local",
      "action_name": "read_web_page_config",
      "local_action_type": "read",
      "file_name": "page_description_config.json",
      "attach_to_site": "facebook"
    }
  ]
}
```

Рисунок 3.7 – Приклад дії ланцюгу конфігураційного файлу

До нелокальних типів дій, що виконуються**,** можна віднести наступні:

– дія**,** виконувана за допомогою бібліотеки Selenium**;**

– дія**,** виконувана за допомогою бібліотеки BeautifulSoup.

Для запуску виконання будь-якої з типів дій функція *execute_config_action* першочергово викликає іншу функцію – *get_config_action*, яка повертає необхідний вузол конфігураційного JSON-файлу (рис. 3.8).

```
def get_config_action(action_name: str):
    for act_el in web_config_obj["actions_chains"]:
        if act_el["action_name"] == action_name:
            return act_el
    return None
```

Рисунок 3.8 – Отримання опису «Дії» з конфігураційного файлу

Після отримання десерілізованого об’єкту JSON-вузла, у функції *execute_config_action* перевіряється значення властивості “*action_executor*” – отриманого об’єкту, в залежності від якої обрана дія буде виконана за допомогою відповідної бібліотеки.

Так, наприклад, дія авторизації має у конфігураційному файлі значення властивості “*action_executor*”, встановлене як “*selenium*”, та сама дія складається, з опису трьох послідовних кроків, які буде повинна виконати ця бібліотека: заповнення поля адреси електронної пошти, заповнення поля з паролем від

облікового запису та натискання кнопки “Увійти” (дія відображена на рис. 3.9), для авторизації у соціальній мережі:

Рисунок 3.9 – Приклад «Дій» для різних виконувачів

Для виконання дії, для якого у ролі виконавця виступає бібліотека Selenium, використаємо функцію *execute_selenium_action,* яка приймає у якості параметрів ім’я дії та список піддій, які треба виконати (рис. 3.10).

```
def execute_selenium_action(action_name, actions_chains_action):
    elements = selenium_get_page_elements(action_name, actions_chains_action)
    if elements:
        for el in elements:
            make_action_with_element(el[0], el[1]["action"], el[1]["value"])
```

Рисунок 3.10 – Використання бібліотеки Selenium

Для виконання дії за допомогою Selenium необхідно перш за все отримати відповідні елементи вебсторінки, з якими буде відбуватися взаємодія. Саме за це відповідає функція *selenium_get_page_elements*, яка викликається у функції *execute_selenium_action* (рис. 3.11).

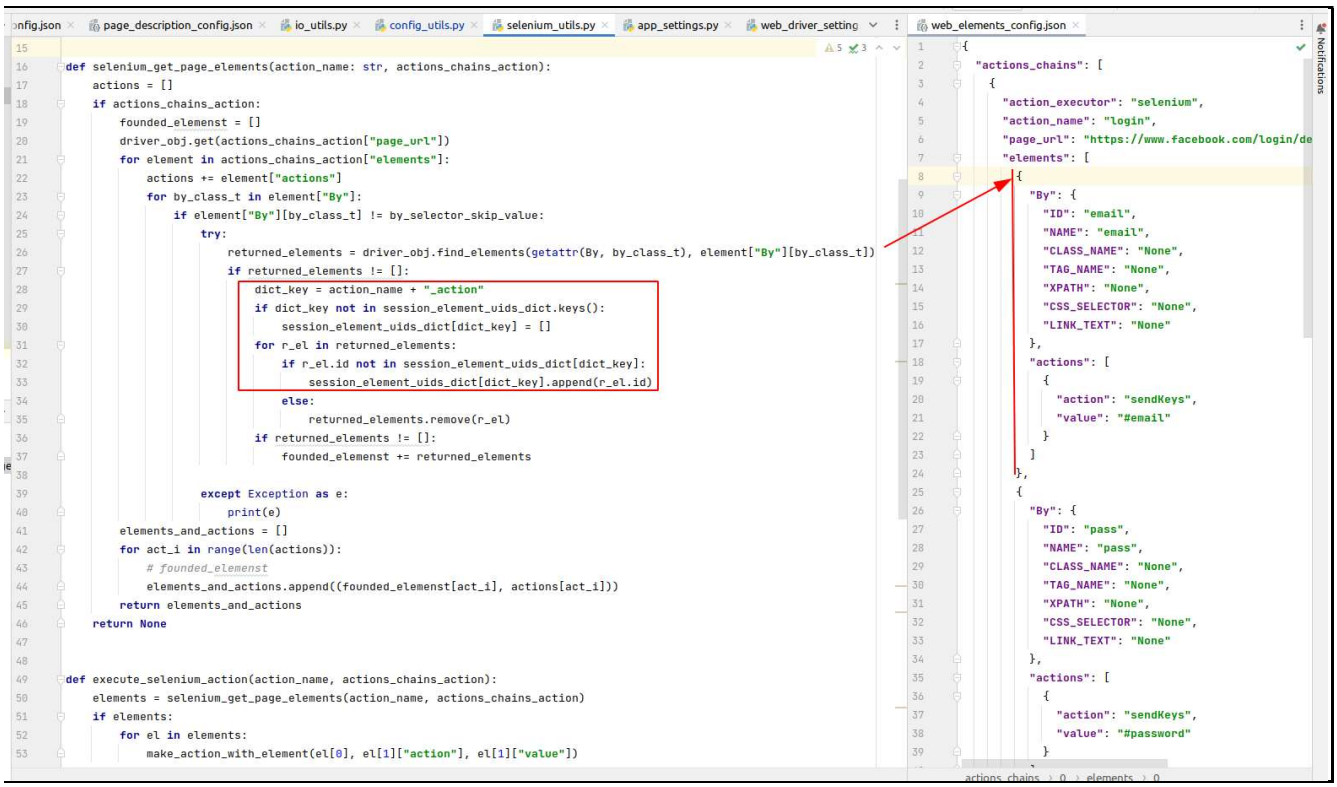


Рисунок 3.11 – Перевірка унікальності зазначеної дії у списку усіх наявних

Функція *selenium_get_page_elements* – шукає на вебсторінці елементи за певними критеріями та значеннями, які повинні приймати критерії пошуку, для того щоб потрапити у список підходящих вебелементів. Імена критеріїв та значення, які вони повинні приймати, також задаються у зовнішньому конфігураційному файлі, при створенні ланцюга дії – як елемент опису виконуваної дії. Для уникнення дублювання, також виконується процедура фільтрації знайдених елементів на унікальність.

Варто зазначити, що перелік імен атрибуту опису дії та значення, які вони повинні приймати, відрізняються для кожного типу виконуваних дій.

Після того, як було знайдено шукані елементи, які відповідають заданим умовам, функція *execute_selenium_action* виконує задану в конфігураційному файлі

дію з певним елементом, підставляючи, за потреби, вказане значення, наприклад, для заповнення текстових форм (рис. 3.12–3.13).

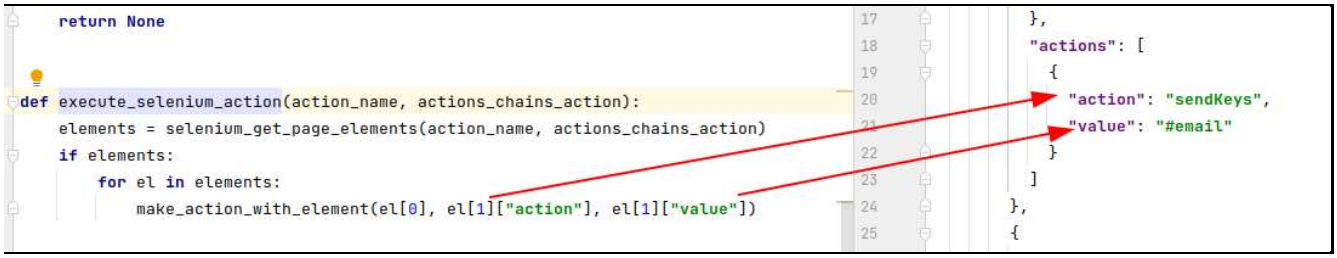


Рисунок 3.12 – Приклад автозаповнення форм

Саме таким чином працює механізм виконання іменованих дій. Після виконання дій авторизації ("*login*"), автоматичної прокрутки сторінки та збору посилань на користувацький профілі у секції коментарів ("*get_posts*") та зчитування додаткового конфігураційного файлу з описом елементів, які підлягають аналізу на вебсторінці ("*read_web_page_config*") – для кожного знайденого посилання на користувацький профіль викликається функція *select_page_profile*. Функція *select_page_profile* грає важливу роль для отримання якісного набору інформації з профіля.

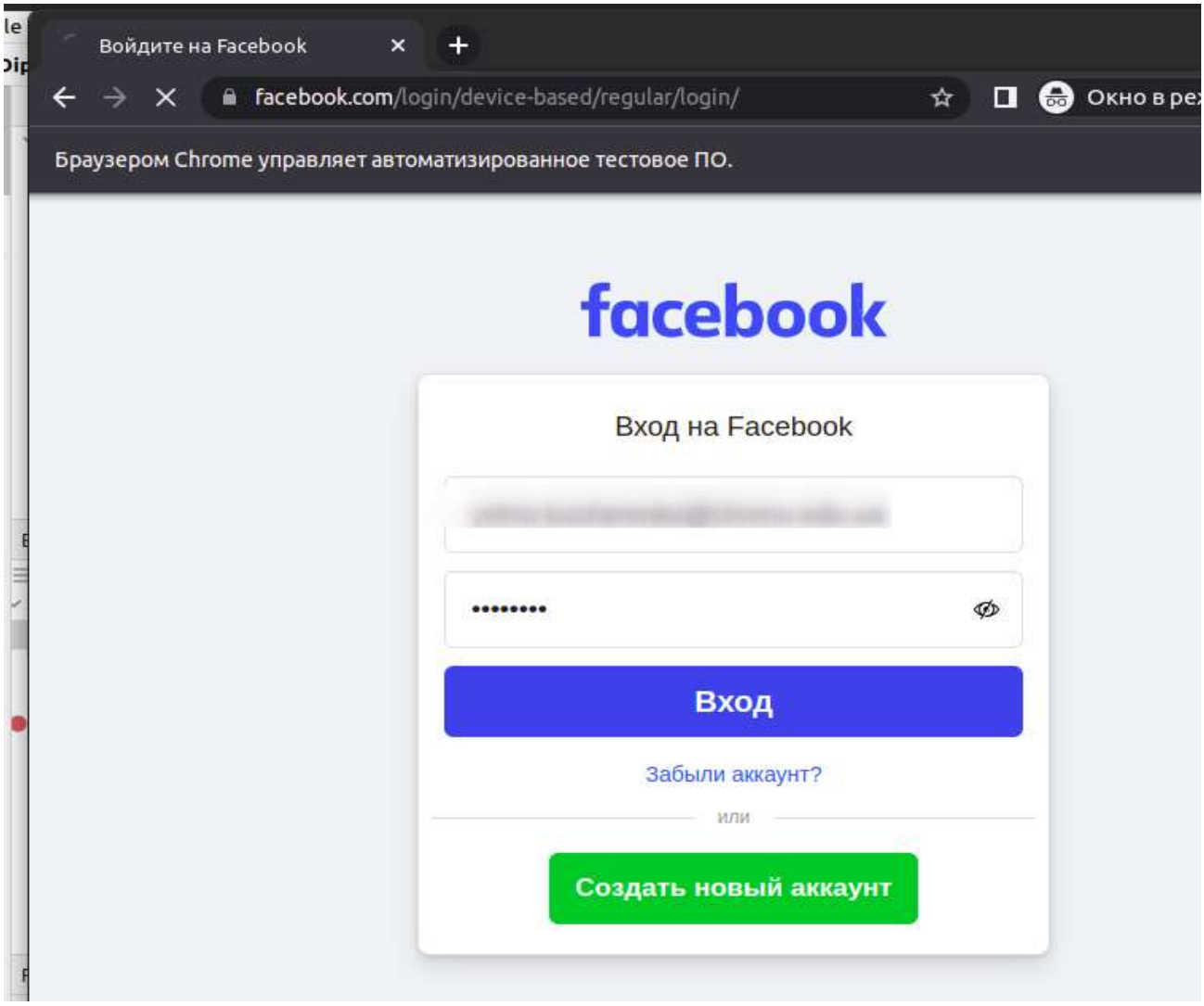


Рисунок 3.13 – Приклад автозаповнення форм

Так, наприклад, для соціальної мережі Facebook можна виділити три основні типи адрес та типів сторінок:

а) найбільш розповсюджений на момент створення системи – формат адреси профілю (рис. 3.14–3.15);

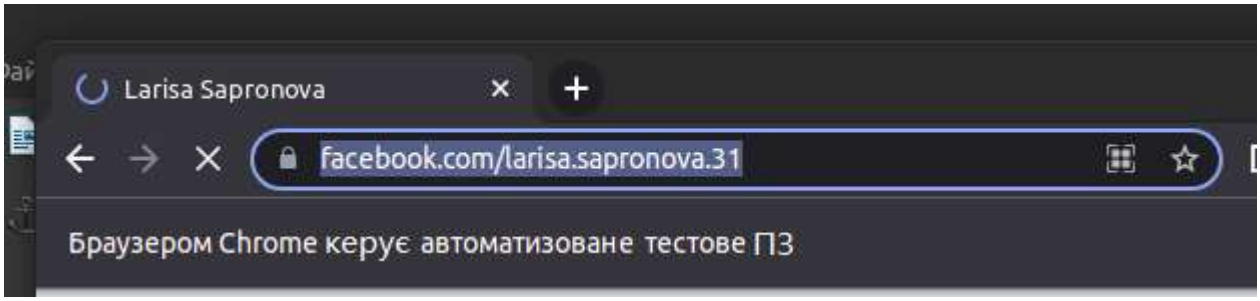


Рисунок 3.14 – Сучасний формат посилання на профіль

Рисунок 3.15 – Застарілий формат посилання на профіль

Застарілий формат посилання на профіль зразка:

profile.php?id=<id користувача>&sk=

б) офіційна сторінка організації чи установи з окремим інформаційним блоком (рис. 3.16).

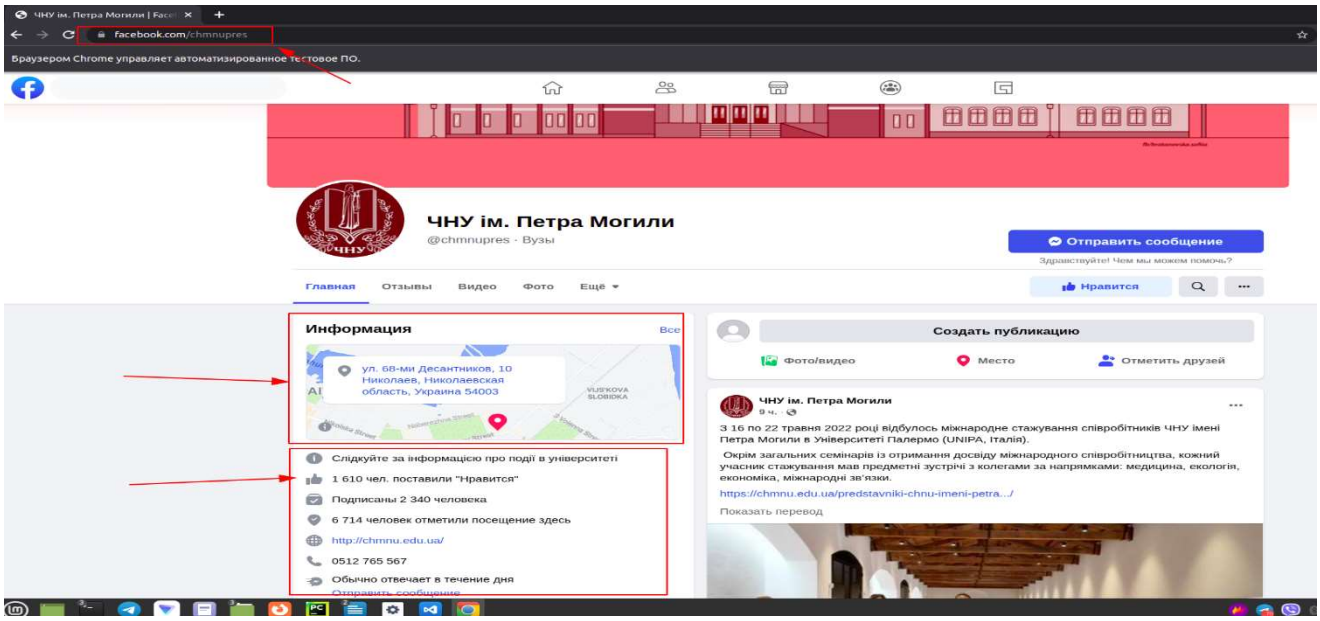


Рисунок 3.16 – Формат посилання на профіль сторінки офіційної організації

Рисунок 3.17 – Приклад налаштування пошукового профілю

Функція *select_page_profile* дозволяє автоматично обирати один з декількох профілів для сторінки, засновуючись на певних признаках або ключових словах, причому, пошук ключового слова можна виконувати не тільки у адресі вебсторінки, але й у її змісті, що також можна вказати при створенні конфігураційного файлу (рис. 3.18).

```
def recursive_profile_search(start_link: str, recursion_depth, social_network_name, page_profile_name, entity_list=None,
                             inherit_entity_list: bool = False):
    profiles_d = []
    try:

        if start_link.endswith("/"):
            start_link = start_link[:-1]
        profile_info = analyze_fb_profile_info(start_link, web_page_config, social_network_name, page_profile_name, "",
                                               entity_list)
        profiles_d.append(profile_info)
        found_friends_profiles = []
        found_family_profiles = []
        if not entity_list or ("/friends" in entity_list or "/about_family_and_relationships" in entity_list):
            if not entity_list or (entity_list and "/friends" in entity_list):
                found_friends_profiles = get_dict_value("friends_#hrefs", profile_info)
            elif not entity_list or (entity_list and "/about_family_and_relationships" in entity_list):
                found_family_profiles = get_dict_value("family_#hrefs", profile_info)
            friends_family_search_list = []
            if found_friends_profiles:
                friends_family_search_list += found_friends_profiles
            if found_family_profiles:
                friends_family_search_list += found_family_profiles
            friends_family_search_list = filter_list_to_unique(friends_family_search_list)
            if not inherit_entity_list:
                entity_list = None

            friends_family_search_list = [lnk for lnk in friends_family_search_list if "photo/" not in lnk]
            for l in friends_family_search_list:
                if l not in get_dict_keys(profiles_d) and (l not in start_link):
                    try:
                        if l.endswith("/"):
                            l = l[:-1]
                        profile_name = select_page_profile(l)
                        profile_info = analyze_fb_profile_info(l, web_page_config, "facebook", profile_name,
                                                               start_link, entity_list)

                        profiles_d.append(profile_info)
                        time.sleep(1)
                    except Exception as e:
                        print(e.__traceback__)
                        print(e)
            if recursion_depth < MAX_RECURSION_DEPTH and len(profiles_d) > 0:
                for p in profiles_d:
                    if p and (locked_found_profile_title not in list(p.values())[0]):
                        recursion_depth += 1
                        profiles_d += recursive_profile_search(start_link=list(p.keys())[0],
                                                               recursion_depth=recursion_depth,
                                                               social_network_name=social_network_name,
                                                               page_profile_name=page_profile_name,
                                                               entity_list=entity_list,
                                                               inherit_entity_list=inherit_entity_list)
            return profiles_d
    except (Exception, KeyboardInterrupt) as ex:
        return profiles_d
    else:
        return profile_info
```

Рисунок 3.18 – Функція *recursive_profile_search*

Так, для двох різних профілів буде відрізнятися значення властивості “*profile_subpages*” – множина ключових слів для пошуку, які відрізняються присутністю для різних видів сторінок.

```
def select_page_profile(search_lnk):
    profile_name = ""
    driver_obj.get(search_lnk)
    bs_obj = bs4.BeautifulSoup(driver_obj.page_source, "html.parser")
    for pr_n, pr_k_w in web_page_config[social_network_name].items():
        selected_keyword = pr_k_w["profile_select_keyword"]
        if pr_k_w["keyword_in"] == "link":
            if selected_keyword != "" and selected_keyword in search_lnk:
                profile_name = pr_n
                break
        elif pr_k_w["keyword_in"] == "page_text":
            print(bs_obj.text)
            official_profile_found_text = bs_obj.find_all(lambda tag: selected_keyword in tag.text)
            if official_profile_found_text:
                profile_name = pr_n

    if profile_name == "":
        profile_name = "default"
    return profile_name
```

Рисунок 3.19 – Функція select_page_profile

У разі, якщо ключова фраза або певна ознака не була знайдена для жодного з варіантів профілей пошуку, обирається профіль за замовчуванням.

Після визначення типу профілю пошуку, для аналізу сторінки користувача використовується рекурсивний метод ***recursive_profile_search***, який приймає наступні параметри: *start_link* – посилання на профіль, який необхідно проаналізувати, *recursion_depth* – глибина рекурсивного обходу пов'язаних профілів, *social_network_name* – назва соціальної мережі, яка підлягає аналізу , *page_profile_name* – ім'я обраного пошукового профілю; та необов'язкові параметри – *entity_list* – список сутностей, які мають бути зібрані при аналізі сторінки; за замовчуванням, даний параметр має значення *None*, при якому процесу збору та аналізу підлягають усі сутності, які перераховані у додатковому конфігураційному файлі *page_description_config*, пошук яких відбувається за допомогою поставлених у відповідність словників з ключовими словами для пошуку; *inherit_entity_list* – булева змінна, яка відповідає за те, чи необхідно наслідувати значення параметру *entity_list* при виконанні рекурсивного пошуку.

При роботі дана функція виконує наступні послідовні дії:

1) викликає функцію ***analyze_fb_profile_info***, передаючи її у якості одного з параметрів посилання на профіль, який необхідно проаналізувати;

2) після виконання первинного аналізу профілю та отримання результатів виокремлює знайдений список посилань на користувацькі профілі друзів та родичів, які були вказані у проаналізованому профілю;

3) для кожного посилання зі списку, отриманого на кроці № 2, та які відповідають певним умовам фільтрації посилань задля перевірки, що це саме посилання на користувацький профіль, знов викликається метод ***analyze_fb_profile_info***;

4) якщо лічильник глибини рекурсивного пошуку менший за зазначений максимум – відбувається рекурсивний виклик методу ***recursive_profile_search***, інакше – скинути лічильник рекурсії до значення, яке було передано при першому виклику функції – у якості параметру та перейти до кроку 3.

Для завершення опису роботи системи розглянемо роботу функції ***analyze_fb_profile_info.***

Перш за все, дана функція виконує перевірку, чи не аналізувався даний користувацький профіль раніше, перевіряючи його наявність у множині текстових рядків – *already_analyze_profiles*. Якщо дана умова виконується – отримає словник ключових слів, за яким необхідно виконувати пошук інформації для заданої сутності на сторінці (рис. 3.20).

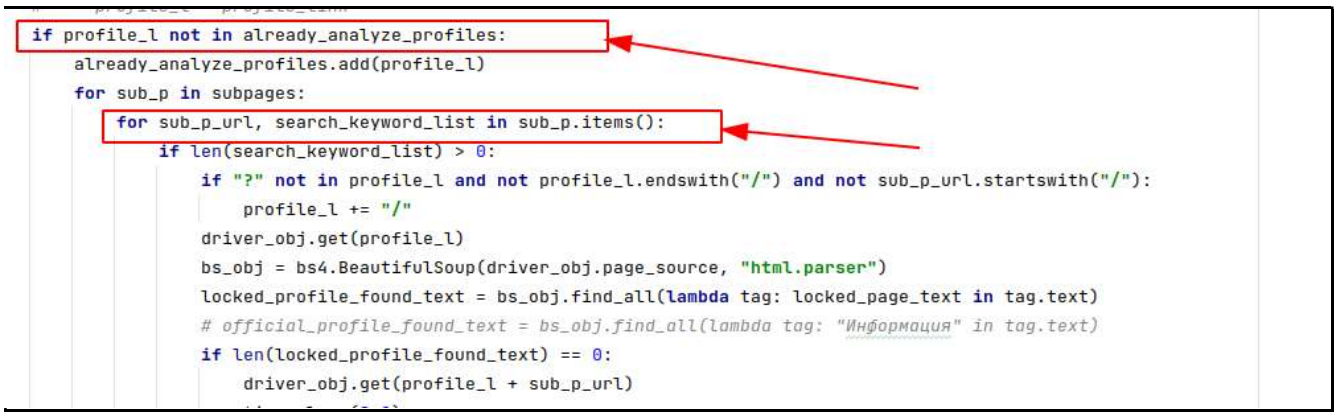


Рисунок 3.20 – Приклад перевірки умови перед аналізом посилання на користувацький профіль

Після отримання словника із конфігураційного файлу відбувається запит та отримання вебсторінки заданого користувацького профілю за посиланням та перевірка умови, що сторінка не відноситься до категорії приватних, тобто таких, на яких уся інформація заблокована для пересічних користувачів мережі та доступна тільки друзям чи підписникам особи. Перевірка даної умови виконується за допомогою пошуку певних константних текстових значень – слів та словосполучень – на сторінці. Якщо сторінка не є приватною – відбувається її подальший аналіз.

Усі сутності, пошук яких відбувається на вебсторінці, можна умовно поділити на чотири групи (рис. 3.21):

```
            for search_keyword in search_keyword_list:
                person_info_dict[search_keyword] = set()
                if "#hrefs" in search_keyword:
                    # if search_keyword == "family_#hrefs":
                    #     print("d")
                    found_urls = get_all_hrefs(bs_obj)
                    filtered_keys = get_config_subpages(subpages)
                    filtered_keys += ["/marketplace", "/groups", "/gaming", "/bookmarks", "/notifications",
                                      "/me",
                                      "/about", "/friends", "/videos",
                                      "/about_overview", "/about_details", "/about_work_and_education",
                                      "/about_life_events", "/about_contact_and_basic_info", "/following",
                                      "/about_places"]
                    filtered_urls = filter_urls(found_urls, filtered_keys, profile_l)
                    filtered_urls = [f_url for f_url in filtered_urls if profile_link not in f_url]
                    while "/" in filtered_urls:
                        filtered_urls.remove("/")
                    person_info_dict[search_keyword] = filtered_urls

                elif "#date" in search_keyword:
                    pass
                elif search_keyword.startswith("/"):
                    person_info_dict[search_keyword] = get_all_hrefs(bs_obj, search_keyword)
                else:
                    info = bs_obj.find_all(lambda tag: search_keyword in tag.text)
                    if info:
                        for info_el in info:
                            person_info_dict[search_keyword].add(info_el.getText())
                    # else:
                    #     person_info_dict[search_keyword] = ""
                person_info_dict[search_keyword] = filter_list_to_unique((person_info_dict[search_keyword]))
                # adding clear
                if not "#" in search_keyword and not "/" in search_keyword:...
        else:
            return {profile_link: locked_found_profile_title}
time.sleep(page_load_delay_secs)
```

Рисунок 3.21 – Приклад різних типів інформації для подальшого пошуку

1) посилання на фото, відео або інші медіа файли, а також на інші користувацькі профілі;

2) дати;

3) адреси інших вебсторінок – пошук та збір посилань, але які не відносяться до множини посилань на медіафайли або сторінки інших користувачів, що є адресами інших сторінок з заданим шаблоном посилання, яке вказується у конфігураційному файлі, наприклад, як наведено на рис. 3.22;

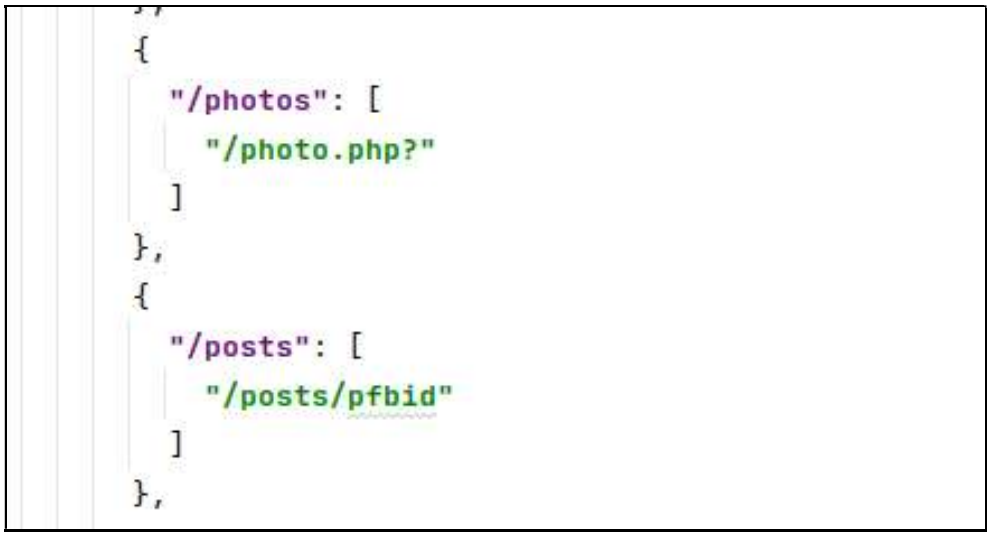


Рисунок 3.22 – Приклад налаштування правил пошуку вебсторінок

4) пошук звичайних текстових даних на сторінці за ключовими словами з словника для пошуку. Після чого відбувається процес пошуку та виокремлення заданої інформації з вебсторінки за допомогою засобів бібліотеки *BeautifulSoup*, її упорядкування та повернення у якості результату роботи програми чи накопичення до списку результатів у разі виконання рекурсивного обходу сторінок (рис. 3.23).

3.3. Демонстрація та опис результатів роботи системи


```
[
 {
  "https://www.facebook.com/irina.tangarova": {
   "Работа": [
    "РаботаНет рабочих мест для показа",
    "Работа и образование",
    "РаботаНет рабочих мест для показаВузИзучала Химический факультет в Университет им. Т.Г.
ШевченкаГод выпуска 1987Средняя школаУчилась в МУ г.БердянскГод выпуска 1982"
   ],
   "Отпуск": [],
   "Живет": [
    "Живет в г. Николаев"
   ],
   "Из г": [
    "Из г. Бердянск"
   ],
   "Развод": [],
   "Брак": [],
   "Мобильный": [],
   "Вуз": [
    "ВузИзучала Химический факультет в Университет им. Т.Г. ШевченкаГод выпуска 1987"
   ],
   "Школа": [],
   "Проживание": [],
   "Проживания": [],
   "Родной": [
    "Родной город"
   ],
```


Рисунок 3.23 – Зміст вихідного файлу програмі

Після завершення роботи програми, у якості результату користувач отримує вихідний файл у форматі JSON з повним переліком інформації, яку вдалося здобути з вебсторінки для кожного з проаналізованих профілів.

Також користувач отримує статистичні чисельні данні для кожної з зібраних сутностей (рис. 3.24).

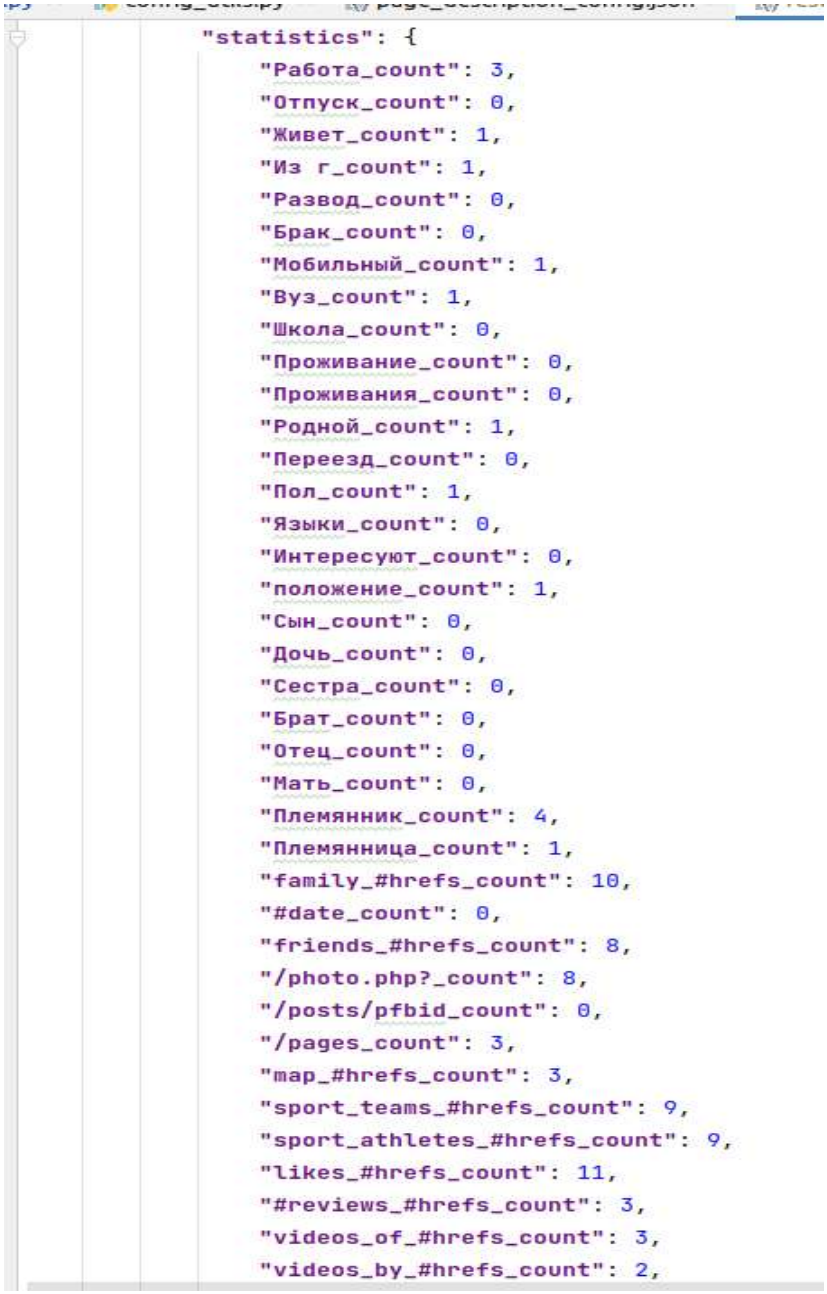


Рисунок 3.24 – Статистичні чисельні дані, отриманні в результаті аналізу профіля

Ці значення допоможуть у подальшому прийняти рішення, за допомогою відповідальної особи (англ. *decision maker*) або експертної системи, щодо профіля – чи є він частиною ворожої системи flood&flame-атак.

Також, для більш зручної роботи користувач отримає графічне представлення залежностей проаналізованих користувацьких профілей один-від-одного, що надасть можливість для більш точного аналізу та оцінки кожного з

профілей на підставі кількості його зв’язків з іншими профілями соціальної мережі (рис. 3.25).

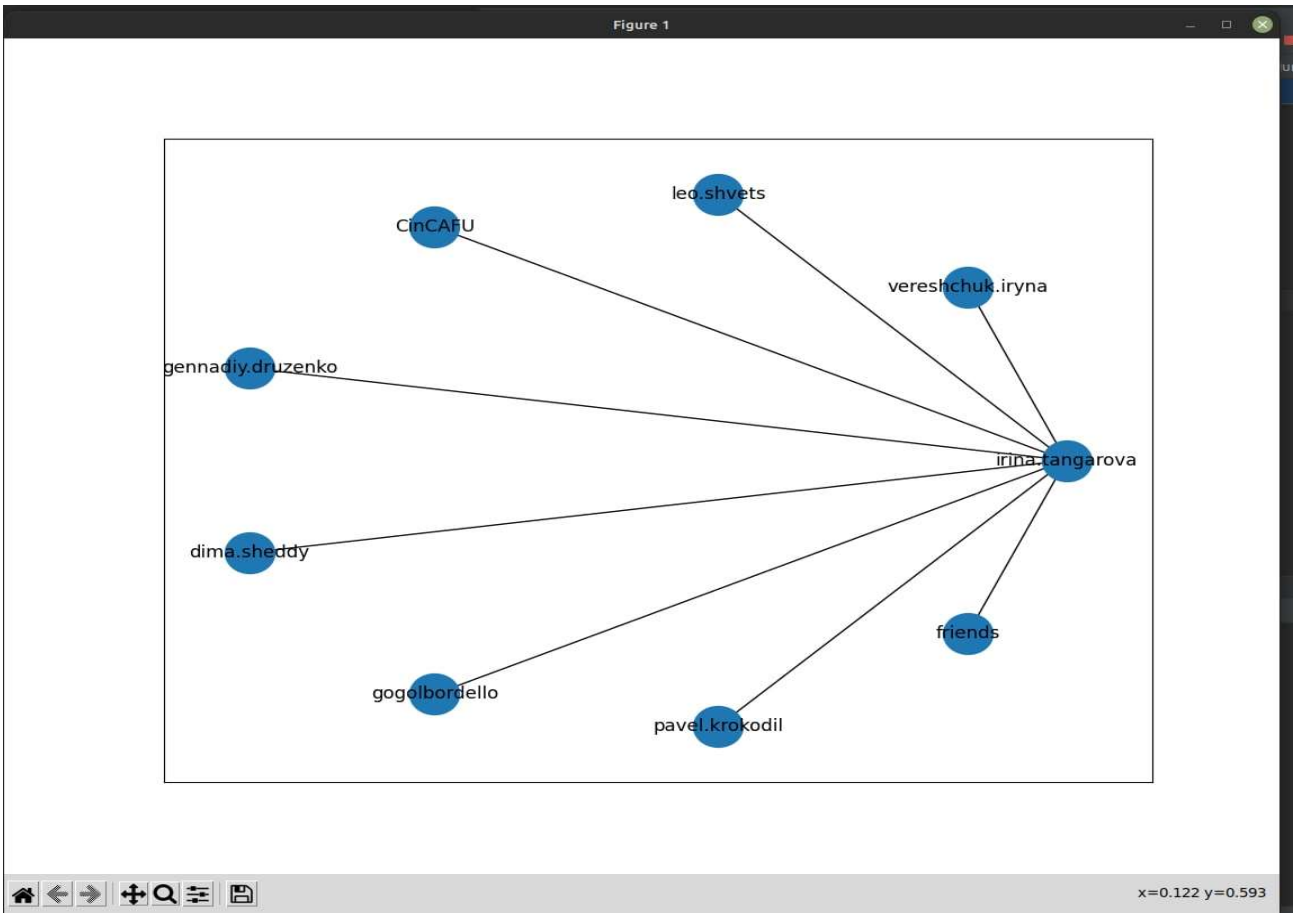


Рисунок 3.25 – Графічне представлення взаємозв’язків знайдених та проаналізованих користувацьких профілів

Графічне представлення знайдених взаємозв’язків являє собою неповно зв’язаний граф, а у деяких випадках його підмножину – дерево, вершиною якого є профіль соціальної мережі з найбільшою кількістю зв’язків.

Висновки до розділу 3

При розробці наведеної системи у її основу було покладено принцип ієрархічних правил для яких визначено терміни – «Дія» та «Виконавець». Саме таке архітектурне рішення дозволило створити або змінювати наявні ієрархічні правила з мінімальними часовими та/або фінансовими витратами.

Особливості технічної реалізації проєкту також дали змогу ефективно поєднати декілька підсистем збору даних у єдину систему та налаштувати їх взаємозв’язок, комбінуючи переваги кожної з них.

Ще одним значним аргументом для використання зазначеного стилю проєктування стало рішення зробити незалежною підсистему візуалізації результатів від роботи основної частини системи. Такий підхід надає змогу виконувати процес візуалізації для демонстрації роботи не тільки у повному циклі роботи програми, але й окремо – виконуючи побудову графу зав’язків – для вже зібраних та оброблених даних.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ БОТ-АТАК НА ОСНОВІ АНАЛИЗУ КОРИСТУВАЦЬКИХ ПРОФІЛЕЙ У СОЦІАЛЬНИХ МЕРЕЖАХ»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810316

Виконав студент 4-го курсу, групи 402

Є. А. Кучеренко

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Консультант *канд. техн. наук*

(наук. ступінь, вчене звання)

А. О. Алексєєва

(підпис, ініціали та прізвище)

«___» _____ 2022 р.

Миколаїв – 2022

4 АНАЛІЗ ВИМОГ ДО ОРГАНІЗАЦІЇ ЗАХОДІВ ТЕХНІКИ БЕЗПЕКИ ТА ОХОРОНИ ПРАЦІ ПРИ РОБОТІ З КОМП'ЮТЕРНОЮ ТЕХНІКОЮ

При роботі з електронним та електротехнічним обладнанням – людина стає частиною виробничого ланцюга – машина-людина, в якому завжди присутній ризик виникнення небезпечних або травматичних для людини ситуацій. Для уникнення факторів, які можуть загрожувати людині під час робочого процесу – необхідно належно забезпечувати виконання умов праці з боку підприємства, до яких відносяться такі важливі фактори, як: безпека на робочому місці, відповідність нормам ДСТУ приміщення, у якому виконуються робота – відносна та абсолютна вологість повітря, інтенсивність, вид та направленість освітлення. Важливим фактором забезпечення умов праці є дотримання санітарних норм щодо урахування впливу електромагнітних та електростатичних полів, які можуть спричинити наступні симптоми погіршення загального відчуття працівника: пригнічення центральної нервової системи (уповільнення реакції, погіршення пам'яті, депресії різної тяжкості, підвищена збудливість, дратівливість, порушення сну, безсоння, різкі перепади настрою, запаморочення, слабкість) – особливо у відносно невеликих офісних приміщеннях, де є висока концентрація робочої офісної техніки. Стандарти та нормативні норми праці для працівників, які працюють за комп'ютером, можуть відрізнятись в залежності від професійних обов'язків та робочого графіку працівника, так, наприклад, офісні працівники постійно контактують з апаратним забезпеченням на своєму робочому місці, а адміністратор серверного парку знаходиться у безпосередній близькості до серверного обладнання, лише у моменти його технічного обслуговування. Саме тому важливо запроваджувати частково індивідуальні норми для окремих верств працівників, які узгоджені з нормами ДСТУ.

4.1 Нормативні правила та вимоги ВООЗ при роботі у офісному приміщенні

При взаємодії людини та апаратно технічного забезпечення, зокрема мобільних та стаціонарних пристроїв індивідуального використання – таких, як мобільних телефонів, ноутбуків та стаціонарних персональних комп'ютерів – кожна людина опосередковано відчуває на собі вплив електромагнітного поля, яке проявляє свою активність при роботі електротехнічного обладнання.

Критерії впливу на людину**:**

1) електромагнітні хвилі призводять до несприятливих змін в організмі, що супроводжуються: пригніченням центральної нервової системи (уповільнення реакції, погіршення пам'яті, депресії різної тяжкості, підвищена збудливість, дратівливість, порушення сну, безсоння, різкі перепади настрою, запаморочення, слабкість);

2) у серцево-судинній системі (зниження ЧСС, зміни на ЕКГ, артеріального тиску);

3) порушення морфологічного складу крові (зменшення кількості лейкоцитів, ретикулоцитів, ацидофільних гранулоцитів), що супроводжується порушеннями функціонального стану ендокринної системи, обмінних процесів, дистрофічними процесами у тканинах мозку, печінці, селезінці, яєчках.

Способи уникнення шкідливого впливу ЕМІ:

– не групувати електроприлади в одному місці, розподілити їх так, щоб вони не посилювали ЕМП один одного;

– не розташовувати електроприлади поряд з обіднім, робочим столом, місцями відпочинку, сну;

– дитяча кімната підлягає ретельному моніторингу на предмет джерел ЕМІ, не допускайте, щоб у ній постійно знаходилися радіокеровані або електричні іграшки, планшет, смартфон, ноутбук;

– включати електроприлади по черзі, а не всі разом: мобільний телефон, комп'ютер, мікрохвильова піч, телевізор повинні працювати в різний час;

– база радіотелефону створює навколо себе стабільне магнітне поле в радіусі 10 метрів, приберіть її зі спальні та робочого столу.

4.2 Умови праці працівників, які використовують у роботі персональні комп’ютери та екрані пристрої

Діяльність більшості працівників сучасних професій у виробничій сфері пов’язана з використанням комп’ютерної техніки. Комп’ютер для сучасної людини є такою ж технічною необхідністю, як телевізор або холодильник. Побутові прилади ми використовуємо не задумуючись про їх шкідливість або нешкідливість, усвідомлюючи лише переваги їх наявності. Щодо комп’ютерів, то існує багато інформації як про їх безпечність, так і шкідливість.

Наказом Держгірпромнагляду від 26.03.2010 р. № 65 затверджено Правила охорони праці під час експлуатації електронно-обчислювальних машин (далі– Правила № 65) [1]. Вони поширюються на всіх суб’єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов’язану з електронно-обчислювальними машинами (ЕОМ) з відеодисплейними терміналами (ВДТ), у т. ч. на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і периферійними пристроями (ПП).

Персональні комп’ютери (ПК) – це теж ЕОМ, тож термін «комп’ютер» і абревіатура «ЕОМ» є синонімами.

Осіб, які працюють з комп’ютерами, поділяють на групи:

а) **розробники програм** (інженери-програмісти) – мають справу переважно з відео терміналами, їх робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті рук;

б) **оператори електронно-обчислювальних машин –** виконують роботу, пов’язану з обліком інформації, одержаної з візуального дисплейного терміналу за

попереднім запитом, або тієї, що самостійно надходить з нього, яка супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи і характеризується як робота з напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у довільному темпі;

в) **оператори комп'ютерного набору** – займаються одноманітною за характером роботою з документацією та клавіатурою, під час якої нечасто та ненадовго переключають погляд на екран дисплея, вводять дані з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті рук на фоні загальної гіподинамії з напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Згідно з визначенням, наведеним у Правилах № 65 , оператор ЕОМ з ВДТ і ПП – працівник, який використовує екранні пристрої під час своєї роботи [2].

4.3 Фактори виробничого середовища і трудового процесу під час роботи з комп'ютером

Працюючи з комп'ютером, працівник потрапляє під вплив різноманітних факторів виробничого середовища та трудового процесу. Розглянемо ці фактори та їх дію на організм людини.

Робочі місця працівників, які працюють з екранними пристроями, мають відповідати ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт. Вони повинні бути спроєктовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення і рухів [3]. Освітлення робочого місця має створювати відповідний контраст між екраном і навколишнім середовищем з урахуванням виду роботи та відповідати ДСанПіН 3.3.2.007-98 [4]. Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями потрібно підтримувати на постійному рівні відповідно до ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень», затверджених постановою МОЗ і Головного державного санітарного лікаря України від 01.12.1999 № 42 [5].

Робочі місця працівників з екранними пристроями не повинні обмежувати рухів персоналу. Робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування [6].

Хімічні речовини у повітрі робочої зони. Хімічний фактор виробничого середовища не є характерним для працівників, які використовують у своїй роботі ПК. Проте, за певних умов він може мати місце. Так, внаслідок інтенсивного використання копіювальної техніки безпосередньо у робочій зоні користувачів ПК може бути присутнім озон (хімічна речовина 1 класу небезпеки, гостроспрямованої дії на організм людини, граничнодопустима концентрація (ГДК) становить 0,1 мг/м3). За результатами проведених гігієнічних досліджень вміст озону, як правило, не перевищує ГДК.

Джерелом хімічних речовин у повітрі робочої зони можуть бути матеріали, які використані для ремонту та оздоблення приміщень. Серед них синтетичні матеріали для підлоги (ворсоніт, ковролін, лінолеум), полімерні матеріали для оздоблення стін, тощо. Найпоширенішою речовиною, яка може виділятися з полімерних матеріалів, є формальдегід (хімічна речовина 2 класу небезпеки, гостроспрямованої та алергенної дії на організм людини, граничнодопустима концентрація 0,5 мг/м3). Вміст формальдегіду у повітрі робочої зони може бути на рівні 0,5 ГДК.

Шум. Нормативним значенням еквівалентного рівня звуку для програмістів є 50 дБА, для операторів у залах оброблення інформації – 65 дБА, для операторів у приміщеннях, де розташовані гучні агрегати – 75 дБА.

За результатами проведених гігієнічних досліджень, еквівалентні рівні шуму на робочих місцях працівників офісних приміщень знаходяться у межах 48-59 дБАекв. Як правило, перевищення рівня шуму на робочих місцях офісних працівників є наслідком телефонних розмов співробітників (у приміщеннях з великою кількістю робочих місць).

Неіонізуюче випромінювання. Як правило, роботу працівників з ПК пов’язують зі шкідливим впливом електромагнітних полів. Це мало місце під час

використання моніторів з електронно-променевою трубкою. У сучасних комп'ютерах використовуються рідкокристалічні монітори, тому вплив електромагнітного поля від монітора практично відсутній. Наявним залишається електростатичне поле на поверхні монітора та поверхні клавіатури [7].

Нормативне значення напруженості електростатичного поля становить 150 В/см. Напруженість електростатичного поля, визначена на поверхні монітора та поверхні клавіатури, як правило, не перевищує нормативного значення. Рівень напруженості електростатичного поля залежить від вологості повітря, регулярного прибирання робочого місця (усунення запиленості).

Норми клімату. У виробничих приміщеннях на усіх робочих місцях з персональними комп'ютерами мають обов'язково забезпечуватися оптимальні значення параметрів мікроклімату, а саме температури відносної вологості й рухливості повітря (СН 4088-86) [8].

Таблиця 4.1 – Норми клімату для приміщень з екранними пристроями

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

Для вимірювання норми було використано гігрометр психрометричний– прилад для вимірювання температури та вологості повітря, простий тип гігрометра. Швидкість випаровування вологи збільшується в міру зменшення відносної вологості повітря. Випаровування вологи, в свою чергу, викликає охолодження конденсованої рідини. Таким чином, температура вологого об'єкта зменшується. За різницею температур повітря і вологого об'єкта можна визначити швидкість

випаровування, а значить, і вологість повітря. При цьому треба враховувати той факт, що волога, яка випаровується, залишається в околицях вологого предмета, і, таким чином, локально збільшується вологість повітря. Для усунення цього ефекту, при вимірюванні вологості, застосовують аспірацію (створюється потік повітря над вологим об'єктом).

В стаціонарних психрометрах термометри закріплені на спеціальному штативі в метеорологічній будці. Основний недолік станційних психрометрів – залежність показів зволоженого термометра від швидкості повітряного потоку в будці. Основний станційний психрометр – психрометр Августа.

Рівні позитивних та негативних іонів в повітрі приміщень з ПК мають відповідати санітарно-гігієнічним нормам № 2152-80.

Таблиця 4.2 – Допустимі норми рівня іонізації повітря приміщень

Рівні	Кількість іонів в 1 см повітря	
	n+	n–
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Штучне освітлення в приміщеннях з робочими місцями ВДТ ЕОМ та ПЕОМ має здійснюватися системою загального рівномірного освітлення, також допускаються додаткове встановлення світильників місцевого освітлення. Зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити в районі 300–500 лк.

Приклад розрахунків норми освітлення для приміщення. Припустимо, існує спальна кімната, розмір якої становить 16 квадратних метрів, а в якості світильника використовується стельова люстра. Згідно із нормою рекомендована кількість світла становить 150 люкс. 150 люкс – це кількість світла в розрахунку на 1 квадратний метр. Тому необхідно помножити 150 люкс на 16 квадратних метрів.

Отримаємо 2,400 люкс – це і буде рекомендована кількість світла для гостьової кімнати. Припустимо, для освітлення були куплені звичайні лампи накопичення – в такому разі доведеться купити 2 лампи по 100 Вт (1.200 * 2 = 2.400 люкс) або 3 лампи по 75 Вт (900 * 3 = 2.700 люкс).

Напруженість електромагнітного поля на відстані 0,5м від будь-якої поверхні відеомонітора не повинна перевищувати гранично допустимих рівнів (ГДР) які наведено у табл. 4.3.

Таблиця 4.3 – Гранично допустимі рівні при роботі з ПК

Діапазон частот	ГДР електричного поля, В/м	ГДР магнітного поля, нТл
5 Гц – 2 кГц	25	250
2 кГц – 400 кГц	2,5	25
3 МГц – 30 МГц	0,25	2,5

Робоча поза. Цей фактор трудового процесу залежить від правильної організації робочого місця для забезпечення зручної робочої пози, адже «закам’яніле» положення шкідливо впливає на хребтово-м’язову систему. Стіл має бути просторим, із спеціальною підставкою для ніг, а робочий стілець мати відрегульовану висоту, певний кут нахилу сидіння і спинки. Під час роботи за комп’ютером людина сидить кілька годин поспіль у незручному положенні. Це не тільки викликає загальну втому, а й може призвести до розвитку остеохондрозу різних ділянок хребта – шийного, грудного, попереково-крижового.

Неправильне положення рук під час друкування на клавіатурі призводить до хронічних захворювань кисті. Тому клавіатура має розташовуватися на відстані 10–15 см від краю столу [9].

Щоб робота за комп’ютером не шкодила здоров’ю, необхідно постійно стежити за своєю поставою. Правильна постава максимально розвантажує м’язи і дозволяє працювати довше, менше втомлюючись.

Напруженість праці. Напруженість праці працюючого з ПК пов’язана, перш за все, з напруженістю очей, тривалістю зосередження уваги, яка може становити понад 75% часу. Тому саме очі найбільш страждають під час роботи з комп’ютером. Велике значення при роботі за комп’ютером мають такі речі як: відстань до екрана, шрифт, розмір тексту на моніторі, наявність або відсутність мерехтіння, яскравість екрана, освітлення робочого місця, наявність перерв у роботі. Саме ігнорування таких простих на перший погляд речей у значній мірі призводить до погіршення зору та хвороб очей.

Висновки до розділу 4

У результаті виконання спеціальної частини з охорони праці було проаналізовано вимоги до організації заходів техніки безпеки, охорони праці під час роботи за комп’ютером та правила організації роботи з електронними пристроями.

Під час роботи за персональним комп’ютером людина швидко втомлюється через значні навантаження на м’язи та хребет, які довго знаходяться у майже нерухомому стані, що спричиняє порушення зоровому апараті та нервовій системі і іноді може призводити до порушень психічного здоров’я людини. Саме тому основні нормативні впровадження, які написанні у ДСТУ є надзвичайно важливими і обов’язковими до виконання під час забезпечення робочого процесу.

На жаль, на сьогоднішній день, не усі умови праці виконуються належним чином. До цього призводять такі фактори, як ненормований робочий графік працівників у деяких випадках, який збільшує загальний час роботи за комп’ютером та іншим офісним обладнанням, а також наявність таких шкідливих факторів, як наприклад шум та шумове забруднення у деяких офісних приміщеннях, що не правильно побудовані або не раціонально використовують виділений простір.

Завадити дії шкідливих умов можуть дисциплінованe виконання вимог та нормативів, що стосуються охорони праці на підприємстві при роботі з

комп’ютерами. Дотримання наведених правил дозволить мінімізувати ризики та можливі негативні наслідки для здоров’я працівника.

ВИСНОВКИ

В результаті виконання дипломної роботи розроблено систему для автоматизованого виявлення бот-атак на основі аналізу користувацьких профілей у соціальних мережах.

В процесі виконання бакалаврської кваліфікаційної роботи досліджено методи програмної та алгоритмічної реалізації цілісної системи для виконання автоматизованого пошуку, перевірки й формування висновків щодо здійснення бот-атак для кожного користувацького профіля у певній соціальній мережі.

Зазначену мету досягнуто завдяки виконанню наступних завдань:

– виконано аналіз типів та методів доступу до інформації з користувацьких профілів шляхом автоматизації;

– виконано аналіз систем захисту збору інформації о користувачах з боку соціальних мереж;

– визначено основні напрямки розробки та певний інструментальний набір технічних засобів для виконання збору та аналізу інформації, зокрема: мови програмування, операційної системи та програмного оточення системи, що розробляється, набір допоміжних зовнішніх бібліотек для прискорення процесу розробки;

– розроблено загальний алгоритм обходу дерева вебсторінок усередині соціальної мережі;

– розроблено основні принципи взаємодії елементів екосистеми збору інформації, а саме – процесу взаємодії технік скрапінгу, парсингу та вебкраулінгу.

Виконання аналізу таких видів поведінки за допомогою сучасних засобів автоматизованих обчислень, заснованих на алгоритмах збору інформації з відкритих джерел, надасть змогу не тільки долучити нові знання про людину як соціальну істоту, але й спрогнозувати її поведінку та перешкодити появі таких видів атак на просторах соціальних мереж або інших систем масових комунікації.

Практична значимість розробленої системи полягає у тому, що за її допомогою можливо виконувати пошук та передчасне знешкодження ворожих бот систем, які спричиняють матеріально-технічні та соціальні збитки.

Робота пройшла апробацію під час Всеукраїнської науково-практичної конференції «Інформаційні технології та інженерія» в ЧНУ ім. Петра Могили (Миколаїв, 8–11 лютого 2022 р.). За результатами конференції опубліковано тези доповіді [10].

Кваліфікаційна робота складається з вступу, трьох розділів, висновку, переліку джерел посилання, двох додатків та спеціальної частини з охорони праці. Основна частина роботи викладена на 68 сторінках тексту (без додатків), містить 6 табл., 27 рис., 20 джерел посилання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mayer E. Spammers show they give a fig about Instagram, launch fruit-based attack. Publ. June 29, 2013. URL: https://www.cnet.com/tech/services-and-software/spammers-show-they-give-a-fig-about-instagram-launch-fruit-based-attack/ (Last accessed: 29.12.2021).

2. Литвиненко Ю. Крупнейшая хакерская атака на Twitter. Опубл. 16 июля 2020 г. URL: https://www.vedomosti.ru/technology/articles/2020/07/16/834712-vzlom-milliarderov-v-twitter (дата обращения: 29.12.2021).

3. The SAGE encyclopedia of the Internet / B. Warf (Ed.). Thousand Oaks, CA, United States : Sage, 2018. 1120 p. DOI: 10.4135/9781473960367.n256. URL: https://sk.sagepub.com/reference/the-sage-encyclopedia-of-the-internet-3v (Last accessed: 29.12.2021).

4. FBI Scrubbed 19,000 PCs Snared By Coreflood Botnet. Publ. June 21, 2011. URL: https://krebsonsecurity.com/2011/06/fbi-scrubbed-19000-pcs-snared-by-coreflood-botnet/ (Last accessed: 29.12.2021).

5. Green J. S. Cyber security: An introduction for non-technical managers. New York, NY, United States : Routledge (Taylor & Francis Group), 2016. 264 p.

6. Бурячок В. Л., Толубко В. Б., Хорошко В. О., Толюпа С. В. Інформаційна та кібербезпека: соціотехнічний аспект : підручник ; за заг. ред. В. Б. Толубка. Київ : Держ. ун-т телекомунікацій, 2015. 288 с.

7. Пропаганда в играх. Типы, виды и методы пропаганды на примере видеоигр и как геймплейная механика. URL: https://www.youtube.com/watch?v=lEASIMjEgZk&t=1468s (дата обращения: 29.05.2022).

8. Как работают видеокодеки? Разбор. URL: https://www.youtube.com/watch?v=gqDuIcJ-alM (дата обращения: 29.05.2022).

9. Как работают библиотеки виртуального окружения. URL: https://habr.com/ru/post/418579/ (дата обращения: 29.05.2022).

10. Ткаченко М. П., Кучеренко Є. А., Журавська І. М. Автоматизована система виявлення бот-атак на основі аналізу користувацьких профілів у соціальних мережах. *Інформаційні технології та інженерія* : тези доп. Всеукр. наук.-практ. конф. Миколаїв, 8–11 лютого 2022 р. Миколаїв : Чорном. нац. ун-т ім. Петра Могили, 2022. С. 28–31.

11. Веб-скрейпинг на PHP. URL: https://habr.com/ru/company/vk/blog/565028/ (дата обращения: 29.05.2022).

12. Shabran R. P., Awasthi U. K. Flame of cyber crimes on social media a burning issue. Booksclinic Publishing, 2021. 176 p.

13. McDarrah T. How to take responsibility: solving Facebook's decision-making divide. Publ. May 17, 2021. URL: https://www.forbes.com/sites/teddymcdarrah/2021/05/17/how-to-take-responsibility-solving-facebooks-decision-making-divide/?sh=6b6961f84441 (Last accessed: 29.12.2021).

14. Johnson K. Facebook AI researchers detect flood and fire damage from satellite imagery. Publ. Dec. 7, 2018. URL: https://venturebeat.com/2018/12/07/facebook-ai-researchers-detect-flood-and-fire-damage-from-satellite-imagery/ (Last accessed: 29.12.2021).

15. Erbschloe M. Social media warfare: Equal weapons for all. CRC Press, 2017. 327 p.

16. Erbschloe M., Vacca J. R. Information warfare: How to survive cyber attacks. Osborne/McGraw-Hill, 2001. 315 c.

17. Libicki M. C. Cyberdeterrence and cyberwar. Rand Corporation, 2009. 238 p.

18. Wang X., Ma X., Peng J., et al. On modeling link flooding attacks and defenses. *IEEE Access*. 2016. Vol. 4. PP:1–20. DOI: 10.1109/ACCESS.2021.3131503.

19. Akib M. S., Antu S. R., Haque B. M. J., et al. Social engineering attack for availability of social media : preprint. Jan. 2021. 25 p.

20. Etuh E., Bakpo F. S., Agozie E. Social media networks attacks and their preventive mechanisms: A review. Jan. 2022. P.59–72. DOI: 10.5121/csit.2021.112405.

ДОДАТОК А

Код програмного забезпечення

Main.py

```
import codecs
import datetime
import json
import time
import bs4
from settings.app_settings import *
from scripts.config_scripts.config_utils import *
from scripts.settings.web_driver_settings import driver_obj
from scripts.additional_utils.gui_utils import create_and_draw_grpah

def get_profiles_links(raw_links: list):
    cut_links = []
    for l in raw_links:
        cut_links.append(l.split("?")[0])
    return cut_links

def get_all_hrefs(bs_obj, search_keyword=None):
    hrefs = []
    info = bs_obj.find_all("a")
    if search_keyword:
        for url in info:
            if url.has_attr("href") and search_keyword in url["href"]:
                hrefs.append(url["href"])
    else:
        for url in info:
            if url.has_attr("href"):
                hrefs.append(url["href"])
    return hrefs

already_analyze_profiles = set()

def get_config_subpages(subpages):
    keys_list = []
    for s_p in subpages:
        keys_list += list(s_p.keys())
    return keys_list

def filter_urls(urls, filter_list, profile_link):
    values_for_remove = set()
    for f_el in filter_list:
        for i in range(len(urls)):
            if urls[i].startswith(f_el) or urls[i].endswith(f_el) or profile_link in urls[i]:
                values_for_remove.add(urls[i])
```

```
    for v in values_for_remove:
        urls.remove(v)
    return urls

all_found_hrefs=set()
def filter_list_to_unique(some_list: list):
    data=list(set(some_list).difference(all_found_hrefs))
    all_found_hrefs.update(data)
    return data

def get_dict_keys(list_of_d: list):
    keys = []
    for d in list_of_d:
        keys += list(d.keys())
    return keys

def analyze_fb_profile_info(profile_link: str, json_config, social_network_name, page_profile_name,
linked_from_profile,
                            req_subpages_list=None):
    person_info_dict = {}
    try:
        subpages = json_config[social_network_name][page_profile_name]["profile_subpages"]
    except Exception as e:
        return None

    if req_subpages_list:
        upd_subpages = []
        for r in req_subpages_list:
            for s_p in subpages:
                if r in s_p.keys():
                    upd_subpages.append(s_p)
        subpages = upd_subpages
    page_profile_template = json_config[social_network_name][page_profile_name]["page_profile"]
    # if not profile_link.startswith("http") and not profile_link.startswith("https"):
    profile_l = page_profile_template.replace("#profile", profile_link)
    # else:
    #     profile_l = profile_link
    if profile_l not in already_analyze_profiles:
        already_analyze_profiles.add(profile_l)
        for sub_p in subpages:
            for sub_p_url, search_keyword_list in sub_p.items():
                if len(search_keyword_list) > 0:
                    if "?" not in profile_l and not profile_l.endswith("/") and not sub_p_url.startswith("/"):
                        profile_l += "/"
                    driver_obj.get(profile_l)
                    bs_obj = bs4.BeautifulSoup(driver_obj.page_source, "html.parser")
                    locked_profile_found_text = bs_obj.find_all(lambda tag: locked_page_text in tag.text)
                    # official_profile_found_text = bs_obj.find_all(lambda tag: "Информация" in tag.text)
                    if len(locked_profile_found_text) == 0:
```

```
            driver_obj.get(profile_l + sub_p_url)
            time.sleep(0.2)
            bs_obj = bs4.BeautifulSoup(driver_obj.page_source, "html.parser")
            for search_keyword in search_keyword_list:
              person_info_dict[search_keyword] = set()
              if "#hrefs" in search_keyword:
                # if search_keyword == "family_#hrefs":
                #     print("d")
                found_urls = get_all_hrefs(bs_obj)
                filtered_keys = get_config_subpages(subpages)
                filtered_keys += ["/marketplace", "/groups", "/gaming", "/bookmarks",
"/notifications",
                                  "/me",
                                  "/about", "/friends", "/videos",
                                  "/about_overview", "/about_details", "/about_work_and_education",
                                  "/about_life_events", "/about_contact_and_basic_info", "/following",
                                  "/about_places"]
                filtered_urls = filter_urls(found_urls, filtered_keys, profile_l)
                filtered_urls = [f_url for f_url in filtered_urls if profile_link not in f_url]
                while "/" in filtered_urls:
                  filtered_urls.remove("/")
                person_info_dict[search_keyword] = filtered_urls

              elif "#date" in search_keyword:
                pass
              elif search_keyword.startswith("/"):
                person_info_dict[search_keyword] = get_all_hrefs(bs_obj, search_keyword)
              else:
                info = bs_obj.find_all(lambda tag: search_keyword in tag.text)
                if info:
                  for info_el in info:
                    person_info_dict[search_keyword].add(info_el.getText())
                # else:
                #     person_info_dict[search_keyword] = ""
              person_info_dict[search_keyword] =
filter_list_to_unique((person_info_dict[search_keyword]))
              # adding clear
              if not "#" in search_keyword and not "/" in search_keyword:
                raw_data_list = person_info_dict[search_keyword]
                if isinstance(raw_data_list, list) or isinstance(raw_data_list, set):
                  cleaned_data = []
                  for d in raw_data_list:
                    if "\n" not in d and user_name not in d and d != search_keyword and \
                        (d.startswith(search_keyword) or d.endswith(search_keyword)):
                      cleaned_data.append(d)
                  person_info_dict[search_keyword] = cleaned_data
          else:
            return {profile_link: locked_found_profile_title}
        time.sleep(page_load_delay_secs)
```

```
        person_info_dict["linked_from_profile"] = linked_from_profile
        result_dict = {profile_link: person_info_dict}
        get_profile_data_statistics(result_dict, profile_link)
        return result_dict

    else:
        return None


def get_dict_value(key_str: str, some_dict: dict):
    found_val = None
    if isinstance(some_dict, dict):
        if key_str in some_dict.keys():
            found_val = some_dict[key_str]
            return found_val
        else:
            for v in some_dict.values():
                found_val = get_dict_value(key_str, v)
                if found_val:
                    return found_val
    elif isinstance(some_dict, list) or isinstance(some_dict, set):
        for s_v in some_dict:
            found_val = get_dict_value(key_str, s_v)
            if found_val:
                return found_val


def get_profile_data_statistics(profile_dat: dict, profile_link):
    keys_names = get_dict_keys([profile_dat[profile_link]])
    profile_dat[profile_link]["statistics"] = {}
    for k in keys_names:
        profile_dat[profile_link]["statistics"][k + "_count"] = len(profile_dat[profile_link][k])


START_RECURSION_DEPTH=-1
def recursive_profile_search(start_link: str, recursion_depth, social_network_name,
page_profile_name, entity_list=None,
                             inherit_entity_list: bool = False):
    global START_RECURSION_DEPTH
    if START_RECURSION_DEPTH==-1:
        START_RECURSION_DEPTH=recursion_depth
    profiles_d = []
    try:

        if start_link.endswith("/"):
            start_link = start_link[:-1]
        profile_info = analyze_fb_profile_info(start_link, web_page_config, social_network_name,
page_profile_name, "",
                                               entity_list)
```

```
    profiles_d.append(profile_info)
    found_friends_profiles = []
    found_family_profiles = []
    if not entity_list or ("/friends" in entity_list or "/about_family_and_relationships" in entity_list):
      if not entity_list or (entity_list and "/friends" in entity_list):
        found_friends_profiles = get_dict_value("friends_#hrefs", profile_info)
      elif not entity_list or (entity_list and "/about_family_and_relationships" in entity_list):
        found_family_profiles = get_dict_value("family_#hrefs", profile_info)
      friends_family_search_list = []
      if found_friends_profiles:
        friends_family_search_list += found_friends_profiles
      if found_family_profiles:
        friends_family_search_list += found_family_profiles
      friends_family_search_list = filter_list_to_unique(friends_family_search_list)
      if not inherit_entity_list:
        entity_list = None

      friends_family_search_list = [lnk for lnk in friends_family_search_list if "photo/" not in lnk]
      for l in friends_family_search_list:
        if l not in get_dict_keys(profiles_d) and (l not in start_link):
          try:
            if l.endswith("/"):
              l = l[:-1]
            profile_name = select_page_profile(l)
            profile_info = analyze_fb_profile_info(l, web_page_config, "facebook", profile_name,
                                    start_link, entity_list)

            profiles_d.append(profile_info)
            time.sleep(1)
          except Exception as e:
            print(e.__traceback__)
            print(e)
      if recursion_depth < MAX_RECURSION_DEPTH and len(profiles_d) > 0:
        for p in profiles_d:
          if p and (locked_found_profile_title not in list(p.values())[0]):
            recursion_depth += 1
            profiles_d += recursive_profile_search(start_link=list(p.keys())[0],
                                    recursion_depth=recursion_depth,
                                    social_network_name=social_network_name,
                                    page_profile_name=page_profile_name,
                                    entity_list=entity_list,
                                    inherit_entity_list=inherit_entity_list)
      recursion_depth=START_RECURSION_DEPTH
      return profiles_d
  except (Exception, KeyboardInterrupt) as ex:
    return profiles_d
  else:
    return profile_info
```

```
def select_page_profile(search_lnk):
    profile_name = ""
    driver_obj.get(search_lnk)
    bs_obj = bs4.BeautifulSoup(driver_obj.page_source, "html.parser")
    for pr_n, pr_k_w in web_page_config[social_network_name].items():
        selected_keyword = pr_k_w["profile_select_keyword"]
        if pr_k_w["keyword_in"] == "link":
            if selected_keyword != "" and selected_keyword in search_lnk:
                profile_name = pr_n
                break
        elif pr_k_w["keyword_in"] == "page_text":
            print(bs_obj.text)
            official_profile_found_text = bs_obj.find_all(lambda tag: selected_keyword in tag.text)
            if official_profile_found_text:
                profile_name = pr_n

    if profile_name == "":
        profile_name = "default"
    return profile_name


if __name__ == "__main__":
    web_config_obj = read_web_element_config(config_path=path_to_json_config)
    if web_config_obj:
        execute_config_action("login")
        profiles_links = get_profiles_links(execute_config_action("get_posts"))
        web_page_config = execute_config_action("read_web_page_config")
        profiles_data = []
        social_network_name = "facebook"
        try:
            for search_link in profiles_links:
                profile_name = select_page_profile(search_link)
                profiles_data += recursive_profile_search(search_link, 0, social_network_name,
profile_name)
                # profiles_data = recursive_profile_search(test_link4, 0,
["/about_family_and_relationships"], False)
        except Exception as e:
            print(e)
        finally:
            result_file_name = f"result_{datetime.datetime.now()}.json"
            with codecs.open(result_file_name, "w", encoding='utf-8') as f:
                encode_data = json.dumps(profiles_data, ensure_ascii=False, indent=4)
                f.write(encode_data)
            create_and_draw_grpah(result_file_name, "facebook.com")

        # finalize_options()
```

user_consts.py

```
user_email = "someadress@somedomain"
user_login = user_email
user_password = "some_password"
user_name = "some_user_name"

template_replacement_dict = {
   "email": user_email,
   "password": user_password
}
```

app_consts.py

```
template_value_signal_symbol = "#"
by_selector_skip_value = "None"
page_load_delay_secs = 5
locked_page_text = "друзья видят"
locked_found_profile_title="have locked/private profile"
MAX_RECURSION_DEPTH = 3
```

app_settings.py

```
from scripts.additional_utils.file_io_utils.io_utils import get_current_work_directory

path_to_json_config = f"{get_current_work_directory()}/configs/web_elements_config.json"
path_to_json_web_page_config = f"{get_current_work_directory()}/configs/"
```

web_driver_settings.py

```
from selenium.common.exceptions import WebDriverException
from selenium import webdriver
from scripts.additional_utils.file_io_utils.io_utils import get_current_work_directory
import sys
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.remote.webelement import WebElement

current_path = get_current_work_directory()
driver_path = f"{current_path}/webdrivers/chromedriver"

try:
   driver_options = webdriver.ChromeOptions()
   driver_options.add_argument("--incognito")
   driver_obj = webdriver.Chrome(executable_path=driver_path, options=driver_options)
except WebDriverException as binary_ex:
   print(binary_ex.msg)
   print("please install google chrom browser on your computer")
```

selenium_utils.py

```
import time

from scripts.app_consts import *
from scripts.settings.web_driver_settings import *
```

```
session_element_uids_dict = {} # for remove duplicates elements by same action, but different By selector classes


def make_action_with_element(element: WebElement, action: str, action_value: [str, Keys]):
    if action == "click":
        element.click()
    elif action == "sendKeys":
        element.send_keys(action_value)


def selenium_get_page_elements(action_name: str, actions_chains_action):
    actions = []
    if actions_chains_action:
        founded_elemenst = []
        driver_obj.get(actions_chains_action["page_url"])
        for element in actions_chains_action["elements"]:
            actions += element["actions"]
            for by_class_t in element["By"]:
                if element["By"][by_class_t] != by_selector_skip_value:
                    try:
                        returned_elements = driver_obj.find_elements(getattr(By, by_class_t), element["By"][by_class_t])
                        if returned_elements != []:
                            dict_key = action_name + "_action"
                            if dict_key not in session_element_uids_dict.keys():
                                session_element_uids_dict[dict_key] = []
                            for r_el in returned_elements:
                                if r_el.id not in session_element_uids_dict[dict_key]:
                                    session_element_uids_dict[dict_key].append(r_el.id)
                                else:
                                    returned_elements.remove(r_el)
                            if returned_elements != []:
                                founded_elemenst += returned_elements

                    except Exception as e:
                        print(e)
        elements_and_actions = []
        for act_i in range(len(actions)):
            # founded_elemenst
            elements_and_actions.append((founded_elemenst[act_i], actions[act_i]))
        return elements_and_actions
    return None


def execute_selenium_action(action_name, actions_chains_action):
    elements = selenium_get_page_elements(action_name, actions_chains_action)
    if elements:
```

```
        for el in elements:
            make_action_with_element(el[0], el[1]["action"], el[1]["value"])


def finalize_options():
    driver_obj.close()


def scroll_infinite_page(scroll_iterations):
    # https://stackoverflow.com/a/43299513
    SCROLL_PAUSE_TIME = 1

    # Get scroll height
    last_height = driver_obj.execute_script("return document.body.scrollHeight")

    for i in range(scroll_iterations):
        # Scroll down to bottom
        driver_obj.execute_script("window.scrollTo(0, document.body.scrollHeight);")

        # Wait to load page
        time.sleep(SCROLL_PAUSE_TIME)

        # Calculate new scroll height and compare with last scroll height
        new_height = driver_obj.execute_script("return document.body.scrollHeight")
        last_height = new_height
```

config_utils.py

```
import selenium.webdriver.remote.webelement as WebElement
from typing import Tuple
from selenium.webdriver.common.keys import Keys
from scripts.app_consts import *
from scripts.user_consts import *
import json
from scripts.selenium_scripts.selenium_utils import execute_selenium_action
from scripts.bs4_scripts.bs4_utils import execute_bs4_action
from scripts.additional_utils.file_io_utils.io_utils import read_json_file
from scripts.settings.app_settings import path_to_json_web_page_config

web_config_obj = None


def read_web_element_config(config_path: str):
    global web_config_obj
    with open(config_path) as f:
        web_config_obj = json.load(f)
    web_config_obj = replace_template_values()
    return web_config_obj
```

```
def get_config_action(action_name: str):
    for act_el in web_config_obj["actions_chains"]:
        if act_el["action_name"] == action_name:
            return act_el
    return None


def replace_template_values():
    global web_config_obj
    for action_chain_obj in web_config_obj["actions_chains"]:
        if action_chain_obj["action_executor"] == "selenium":
            for element in action_chain_obj["elements"]:
                for action in element["actions"]:
                    if len(action["value"]) > 0 and action["value"][
                        0] == template_value_signal_symbol:  # if value == #<some_val>
                        if action["value"][1:] in template_replacement_dict.keys():  # if <some_val> in
template_rep_d
                            action["value"] = template_replacement_dict[action["value"][1:]]
    return web_config_obj


def execute_config_action(action_name: str):
    actions_chains_action = get_config_action(action_name)
    if actions_chains_action["action_executor"] == "selenium":
        execute_selenium_action(action_name, actions_chains_action)
        return None
    elif actions_chains_action["action_executor"] == "bs4":
        return execute_bs4_action(action_name, actions_chains_action)
    elif actions_chains_action["action_executor"] == "local":
        if actions_chains_action["local_action_type"] == "read":
            return {actions_chains_action["attach_to_site"]: read_json_file(
                path_to_json_web_page_config + actions_chains_action["file_name"])}
```

bs4_utils.py

```
import time

import requests
from bs4 import BeautifulSoup
from scripts.settings.web_driver_settings import driver_obj
from scripts.app_consts import page_load_delay_secs
from scripts.selenium_scripts.selenium_utils import scroll_infinite_page


def execute_bs4_action(action_name: str, actions_list):
    time.sleep(page_load_delay_secs)
    found_data = []
    for el in actions_list["elements"]:
        page_response_data = None
        if el["actions"]["page_request_method"] == "requests":
            response = requests.get(actions_list["page_url"])
```

```
        if response.status_code == 200:
            page_response_data = response.text
    elif el["actions"]["page_request_method"] == "selenium":
        driver_obj.get(actions_list["page_url"])
        time.sleep(page_load_delay_secs)
        if el["actions"]["need_pre_scroll"]["need"] == "yes":
            scroll_infinite_page(el["actions"]["need_pre_scroll"]["scroll_iterations"])

        page_response_data = driver_obj.page_source
    if page_response_data:
        bs = BeautifulSoup(page_response_data, "html.parser")
        if el["actions"]["action"] == "get_all":
            parsed_elements = bs.find_all(el["By"]["CLASS"], recursive=True)
            for hr in parsed_elements:
                element_attrib = el["By"]["ATTRIB"]
                if hr.has_attr(element_attrib) and el["By"]["TEXT"] in hr[element_attrib] \
                        and el["By"]["NOT_TEXT"] not in hr[element_attrib]:
                    found_profile = hr[element_attrib]
                    if found_profile not in found_data:
                        found_data.append(found_profile)
time.sleep(page_load_delay_secs)
return found_data
```

io_utils.py

```
import json
import os

def get_current_work_directory():
    current_path = os.getcwd()
    current_path = "/".join(current_path.split("/")[0:-1])
    return current_path


def read_json_file(config_file_path: str):
    if os.path.exists(config_file_path) and os.path.isfile(config_file_path):
        if config_file_path.endswith(".json"):
            with open(config_file_path, "r") as f:
                config_data = json.load(f)
                return config_data
```

gui_utils.py

```
import math

from scripts.additional_utils.file_io_utils.io_utils import read_json_file
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib.backend_bases import MouseButton


def link_profiles(json_file_path: str):
    json_data = read_json_file(json_file_path)
    linked_profiles = {}
    if json_data:
        for el in json_data:
            k = list(el.keys())[0]
            if k not in linked_profiles.keys():
                linked_profiles[k] = []
        for el in json_data:
            k = list(el.keys())[0]
            linked_from = el[k]["linked_from_profile"]
            if linked_from != "":
                linked_profiles[linked_from].append(k)

        return linked_profiles


graph_obj = None
nodes_coords = []
json_file_path = None
site_n = ""

def on_click(event):
    global graph_obj
```

```
    global nodes_coords
    global json_file_path
    global site_n
    json_data = read_json_file(json_file_path)
    if event.button is MouseButton.LEFT:
        print('disconnecting callback')
        click_y_float = event.ydata
        click_x_float = event.xdata
        click_x, click_y = click_x_float, click_y_float
        print(click_x, click_y)
        if graph_obj:
            for coord_tuple in nodes_coords:
                for n_name, n_coords in coord_tuple.items():
                    node_x = n_coords[0]
                    node_y = n_coords[1]
                    if click_x+0.2 > node_x and (click_y*-1)-0.5 < node_y<click_y+0.5:
                        plt.figure()
                        plt.axis('off')
                        if not site_n.endswith("/") and not n_name.startswith("/"):
                            site_n += "/"
                        n_name = site_n + n_name
                        for el in json_data:
                            if n_name in el.keys():
                                profile_text = str(el[n_name])
                                plt.text(0.5, 0.5, profile_text, fontsize=10)
                        plt.show()


def create_and_draw_grpah(json_file_p: str, site_name):
    global graph_obj
    global nodes_coords
    global json_file_path
    global site_n
    site_n = site_name
    if not site_n.startswith("https:") and not site_n.startswith("http:"):
        if not site_n.startswith("www"):
            site_n = "www." + site_n
        site_n = "https://" + site_n
    if site_n.endswith("/"):
        site_n = site_n + "/"
    json_file_path = json_file_p
    linked_profiles = link_profiles(json_file_p)
    graph_obj = nx.Graph()
    tmp_dict_fir_keys_upd = {}
    for k, v in linked_profiles.items():
        new_values = []
        if site_name in k:
            for vv in v:
                if site_name in vv:
```

```
            if vv.split("/")[-1] != "":
                new_values.append(vv.split("/")[-1])
        tmp_dict_fir_keys_upd[k.split("/")[-1]] = new_values

linked_profiles = tmp_dict_fir_keys_upd
keys_for_remove = []
for k, v in linked_profiles.items():
    if len(v) == 0 and len(linked_profiles) > 1:
        keys_for_remove.append(k)
for i in keys_for_remove:
    linked_profiles.pop(i)
for link_d_el in linked_profiles.keys():
    graph_obj.add_node(link_d_el)

created_edges_pairs = {}
if len(linked_profiles) > 1:
    for k1, v1 in linked_profiles.items():
        for k2, v2 in linked_profiles.items():
            if k1 != k2 \
                    and (k1 not in created_edges_pairs.keys() and k2 not in created_edges_pairs.values()) \
                    and (k2 not in created_edges_pairs.keys() and k1 not in created_edges_pairs.values()) \
                    and k1 in v2:
                graph_obj.add_edge(k1, k2)
                created_edges_pairs[k1] = k2
else:
    for k1, v1 in linked_profiles.items():
        for k2 in v1:
            graph_obj.add_edge(k1, k2)
positions = nx.kamada_kawai_layout(graph_obj, dist=nx.shortest_path_length(graph_obj))
plt.figure(figsize=(9, 9))
nx.draw_networkx(graph_obj,
                 pos=positions,
                 node_size=1000)
plt.connect('button_press_event', on_click)
pos = nx.kamada_kawai_layout(graph_obj)
for node_n, node_el in pos.items():
    nodes_coords.append({node_n: (node_el[0], node_el[1] * -1)})
plt.show()
```

ДОДАТОК Б

Матеріали апробації роботи

Міністерство освіти і науки України
Чорноморський національний університет
імені Петра Могили

PETRO MOHYLA BLACK SEA NATIONAL UNIVERSITY
COMPUTER SCIENCE FACULTY
Est. 1996

«Інформаційні технології та інженерія»

Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів

ТЕЗИ

9–11 лютого 2022 року

Миколаїв – 2022

ЗМІСТ

Інформаційні системи та їх інтелектуалізація

Машинне навчання та штучний інтелект

Розглянувши переваги синтезу 3 технологій, ми отримуємо потужний популярний та вигідний інструмент для вдосконалення вже існуючих продуктів, розширення сфер застосування та вирішення цілої низки проблем. Представленні основні положення та ключові аспекти роботи та їх сильні сторони. Також отримання підвищення рівня безпеки робить можливим використання у вразливих від кібератак речах.

УДК 004.725: 004.056

Ткаченко М. П., Кучеренко Є. А., Журавська І. М.
Чорноморський національний університет імені Петра Могили,
м. Миколаїв, Україна

АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ БОТ-АТАК НА ОСНОВІ АНАЛІЗУ КОРИСТУВАЦЬКИХ ПРОФІЛІВ У СОЦІАЛЬНИХ МЕРЕЖАХ

Починаючи з 2004 року – з появи та розвитку соціальних інтернет-мереж, таких як: «Facebook» (2004 р.), «Twitter» (2006 р.), «Instagram» (2010 р.) та інших – виникла проблема соціально-технічного характеру: неможливість забезпечити технічний захист від деяких видів атак, що відносяться до соціального аспекту функціонування мережі, нп., flame- та flood-атак [1, 2].

Атаки flame.&.flood – це категорія кібератак, що ставлять на меті виконати соціальний та психологічний тиск на обрану ціль. Основою засобом проведення flood-атаки є багаторазове та масове відправлення повідомлень, які можуть мати безглуздий, агресивний або ще будь-який характер, що призводить до потрібної поведінки жертви та у решті-решт до мети, яку обрав зловмисник. Це може бути, наприклад, доведення цілі до стану психозу або іншого психічного розладу у легкій формі з метою шантажу або спричинення шкоди ще й фізичному здоров'ю жертви – наприклад, доведення до самогубства.

Flame-атаки, на відміну від flood-атак, можуть та найчастіше є більш короткочасними, бо мають інший механізм соціального впливу на обрану ціль. Головна ціль flame-атаки – виявлення «гострих» політичних та/або соціальних тем, а також соціальної групи (друзів, родичів, знайомих, колег тощо), до якої входить жертва даного виду атаки. Крім того, метою flame-атаки є формування двох або більше суперечливих думок або складної – з соціально-політичного боку –

28

тези чи висловлювання, що призводить до бурхливого обговорення обраної теми серед учасників соціальної групи. В результаті це може призвести до розпаду такої групи або погіршенню відносин серед її учасників. Варто зазначити, що на відміну від flood-атаки, основною метою flame-атаки є не формування певної думки за допомогою її масового повторення від «різних» джерел. Її головне призначення – це саме першочергове підняття «гострої» теми з її подальшим обговоренням самими учасниками обраної соціальної групи.

Найчастіше описані вище атаки виконуються за допомогою соціальних інтернет-мереж та системи бот-програм – спеціального виду автоматизованого програмного забезпечення, яке імітує соціальну активність реальних користувачів з першочерговою метою викликати довіру та емпатію інших учасників. Такий підхід призводить до більш зацікавленого обговорення теми з їх боку та формування певної думки за допомогою її циклічного повторення від різних екземплярів бот-кластеру.

Одним із способів протидії flame.&.flood атакам є автоматизований аналіз профілів соціальних інтернет-мереж за заданим набором параметрів, Таким чином можливо виявлення потенційних учасників бот-системи та попередження інших учасників про виявлену загрозу маніпулювання їх соціальною поведінкою [3].

Для забезпечення можливості автоматизованого аналізу профілів користувачів необхідно отримати інформацію заданого набору профілів для їх подальшого об'єднання у кластер та розбиття на підкластери на підставі заданих параметрів.

У розроблюваній автоматизованій системі виявлення бот-атак для отримання даних користувацьких профілів запропоновано використовувати «парсинг» – автоматизований аналіз сирцевого коду вебсторінки. За такою технологією можливо відокремлення певних елементів вебсторінки та отримання їх значень. Наприклад, це може бути пошук HTML-тегу *<title>* та отримання назви вебсторінки або будь-якого іншого тегу та елементу вебсторінки і його значення. Також передбачено використання інструментів імітації та автоматизації користувацьких дій, таких як: натискання на віртуальні кнопки та перехід за посиланням на інші сторінки.

За допомогою парсингу та автоматизованого обходу заданого дерева сторінок – з урахуванням глибини та зв'язності дерева (кількості нащадків для певного вузла) можливо здійснити аналіз та прогнозування того, наскільки інформація з профіля, що є вузлом

29

побудованого дерева, є достовірною та не відноситься до елементів бот-системи.

Для виконання поставленої мети використовуються нижчезазначені технології та методи.

Технології:

а) мова програмування Python;

б) додаткові бібліотеки мови програмування Python:

1) *Selenium* – бібліотека для автоматизації імітування взаємодії з графічним користувацьким інтерфейсом (GUI) та його елементами;

2) *BeautifulSoup4* – бібліотека для парсингу вмісту HTML та XML-файлів;

3) *Requests* – бібліотека для виконання HTTP- та HTTPS-запитів;

4) *NetworkX* та *Matplotlib* – бібліотеки для візуалізації графів, дерев та довільних підготовлених масивів даних.

Методи:

а) аналіз часових рядів;

б) кластеризація та кластерний аналіз;

в) засоби та алгоритми Data Mining – алгоритми збору даних у часовому полі, їх аналіз та пошук шаблонів (патернів) та взаємозв'язків між окремими частинами даних. Спроба використання знайдених патернів на інших наборах даних для пошуку прихованих взаємозв'язків;

г) засоби та алгоритми Data Analysis – алгоритми роботи з даними, які на відміну від Data Mining, не призначені для пошуку скритих патернів та можуть використовуватись на неструктурованих або погано/недостатньо структурованих наборах вхідних даних. Але основною задачею цих алгоритмів як раз і є процес впорядкування та організації «сирих» даних для отримання корисної інформації у більш зручному вигляді.

Таким чином, аналіз користувацьких профілів у соціальних мережах може бути ефективно використаний для виявлення бот-атак за допомогою розробленої автоматизованої системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mayer E. Spammers show they give a fig about Instagram, launch fruit-based attack. Publ. June 29, 2013. URL: https://www.cnet.com/tech/services-and-software/spammers-show-they-

30