Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству”

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Мельник І.В.
«____» _____ 2025 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Мельнику Іллі Вадимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн.наук, пофесор*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Мельник І.В.
(прізвище та ініціали)

АНОТАЦІЯ

Мельник І.В. Програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

Метою розробки є програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

Результат роботи – програмна реалізація системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: кібербезпека, Інтернет-шахрайство

ABSTRACT

Melnyk I.V. Software for a browser plugin cybersecurity system to combat Internet fraud. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a browser plugin cybersecurity system to combat Internet fraud.

The purpose of the development is software for a browser plugin cybersecurity system to combat Internet fraud.

The result of the work is a software implementation of a browser plugin cybersecurity system to combat Internet fraud.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: cybersecurity, Internet fraud

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DHCP – протокол, що використовується для динамічного розподілу IP-адрес.

DNS – розподілена служба Інтернет, використовувана для зіставлення логічних (доменних) імен і IP-адрес. DNS використовується для забезпечення можливості роботи зі зрозумілими й іменами, що легко запам'ятовуються, замість IP-адрес у числовому форматі

ICMP – протокол

IDS – система виявлення атак

IP – адресний протокол

LAN – локальна мережа

NAT – трансляція IP-адрес із одного адресного простору в IP-адреси іншого адресного простору

Proxy – програма-посередник, що транслює запити різних протоколів із локальної мережі в зовнішню мережу

TCP – протокол обміну даними на транспортному рівні

UDP – протокол

URL – уніфікований покажчик інформаційного ресурсу (стандартизований рядок символів, що вказує місцезнаходження документа в мережі Інтернет)

ЕЦП – електронний цифровий підпис

КД – ключова дискета

НСД – несанкціонований доступ

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ВСТУП

Актуальність теми. Антифішинг – це набір технологій, використовуваних для захисту від мережного шахрайства й розкрадання особистих даних, так званого фішингу. В основі антифішинга лежить механізм оповіщення Інтернет-користувачів про влучення на підроблені веб-сайти, які спеціально створюються зловмисниками для збору конфіденційних даних (паролів доступу до онлайн-банків, платіжних систем і інших сервісів).

Антифішинг реалізується за допомогою двох взаємодоповнюючих технологій. Перша – це вбудований в Інтернет-браузер плагін (антифішинговий фільтр), що попереджає користувача про влучення на підроблені або підозрілі сайти. Такі плагіни вже вбудовані в багато популярних браузерів (наприклад, Microsoft Internet Explorer 7.0 і вище або Firefox 2.0 і вище) або комплексні продукти по захисту ПК. Друга – це фільтрація фішингових листів, що розсилаються зловмисниками для заманювання жертв на підроблені веб-сайти, за допомогою персонального або серверного антиспаму.

Антиспам або спам-фільтри – це програмне забезпечення для автоматичного визначення й фільтрації небажаних електронних повідомлень (спаму). Антиспам може використовуватися як на персональних домашніх комп'ютерах, так на корпоративних поштових серверах або Інтернет-шлюзах.

У сучасних антиспамах використовуються два основних підходи до фільтрації небажаних повідомлень. Перший полягає безпосередньо в аналізі змісту повідомлення, а другий – в аналізі репутації відправника й формальних ознак, таких як масовість розсилання.

Поряд з ефективністю фільтрації спаму, а вона в багатьох лідируючих серверних продуктах перевищує 95%, важливим і критичним для багатьох клієнтів параметром є кількість помилкових спрацьовувань, коли легітимні повідомлення можуть помилково відзначатися як спам.

Персональні антиспами, як правило, будуються на основі статистичних методів, заснованих на теоремі Байеса. При цьому щораз при одержанні нових повідомлень користувач антиспама може його навчати, тим самим підвищуючи ефективність фільтрації спаму й зменшуючи кількість помилкових спрацьовувань.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем плагіну браузера для протидії Інтернет-шахрайству.

– Дослідження системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

– Програмна реалізація системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі плагіну браузера для протидії Інтернет-шахрайству.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Сьогодні кожний третій користувач Всесвітньої павутини ставав жертвою фішерів. На жаль, даних по Україні немає, але по загальному рівню кіберзлочиності можна припустити, що тут ці цифри не занадто відрізняються від світових.

На щастя, для того, щоб не стати жертвою шахраїв, потрібно лише «включати голову» і дотримувати необхідної обережності, без якої в житті що-небудь зробити взагалі досить важко. Нижче будуть наведені правила, які дозволять вам не потрапити в пастку фішерів при спілкуванні в соціальних мережах.

По-перше, не довіряйте нікому, навіть тим, хто вам давно знаком і хто ніколи б не став вам підсувати посилання на фішинговий сайт. На жаль, крадіжка аккаунта в соціальній мережі сьогодні – справа буквально декількох хвилин, і тому не можна ніколи точно знати, чи дійсно пересилає вам посилання ваш знайомий, або це зловмисник, заволодівши його обліковим записом, намагається досягти з її допомогою своїх, м'яко говорячи, не занадто шляхетних цілей.

По-друге, будьте уважні. Фішинговий сайт завжди хоч мало-мало, але відрізняється від свого «прототипу», і уважна людина може помітити ці розходження. Вони починаються вже з адреси сайту. Якщо сайт має домен odnok1assniki.su або facedook.tk, краще зовсім не переходити по такому посиланню. Також вас повинне насторожити незвичайне поводження браузера. Ще варто звертати увагу на протокол: фішингові сайти не працюють через HTTPS.

По-третє, по можливості, використовуйте замість логіна й пароля SMS-автентифікацію – на щастя, більшість соціальних мереж її давно підтримують.

Жоден фішинговий сайт ніколи не буде намагатися украсти у вас облікові дані для SMS-автентифікації – просто тому що в зловмисників немає доступу до вашого мобільного телефону, щоб таку автентифікацію пройти. Взагалі, ризик втрати доступу до свого аккаунту при використанні SMS набагато нижче, і це стосується не тільки соціальних мереж.

По-четверте, пам’ятайте, що соціальні мережі в розсильних листах ніколи не просять переслати їм дані для входу, нібито для перевірки. Правда, останнім часом число фішерів, що просять переслати їм логін і пароль поштою, сильно зменшилося – видимо, користувачі все-таки стали більш розумними .

Ну, і в-п'ятих, пам’ятайте, що крім фішингу є й інші засоби для одержання ваших логіна й пароля – наприклад, клавіатурні шпигуни. Тому намагайтеся не забувати й про такі речі, як банальний антивірус.

1.2 Область застосування

Дослідження показало, що ризиковане поводження Інтернет-користувачів створило нові можливості для атак кіберзлочинців. От лише деякі результати:

– 65 відсотків користувачів дозволяють своїм браузерам зберігати паролі онлайн;

– 40 відсотків користувачів не використовують паролі для захисту своїх мобільних пристроїв;

– 28 відсотків респондентів дозволяють мобільним додаткам одержувати доступ до своїх профілів у соціальних мережах;

– 10 відсотків користувачів мобільних пристроїв припускають, що завантажили шкідливий додаток на свій пристрій;

– 40 відсотків повідомили, що діляться результатами своїх ігор у соціальних мережах.

Існують наступні п'ять найбільш типових ситуацій, які містять ризики для користувача Інтернет:

– Крадіжка даних.

– Керування паролями.

– Фішинг.

– Завантаження додатків на мобільні пристрої.

– Керування налаштуваннями приватності в соціальних мережах.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Чим більш успішним стає захист від атак на ті або інші уразливості, тим частіше зловмисники розуміють, що сама слабка ланка в обороні – людин. Теоретично допомогти неуважному й безтурботному користувачеві можуть антифішингові технології, застосовувані в сучасних браузерах, тому що фактично першою лінією оборони в цьому випадку виступає сам браузер. Саме такі технології й будуть розглянуті в даному розділі.

Антифішинговий захист в Opera

У цьому браузері для захисту від фішингу використовується функція «Захист від шахрайства» (Fraud and Malware Protection), яка включена за замовчуванням. На початку кожного сеансу з конкретним веб-сайтом вона перевіряє адресу, використовуючи шифрований канал (https): передає ім'я домену й адреса запитуваної сторінки на спеціальний сервер, де шукає його в чорних списках фішингових посилань, формованих Netcraft (www.netcraft.com) і PhishTank (www.phishtank.com), а також у списках сайтів зі шкідливим ПЗ, які веде «Яндекс».

Якщо доменне ім'я збіжиться з іменем із чорного списку, сервер Fraud and Malware Protection поверне браузеру XML-документ, у якому буде описана проблема (фішинг або шкідливе ПЗ).

При цьому необхідно врахувати:

– Opera Fraud and Malware Protection server не зберігає IP-адресу користувача або будь-яку іншу ідентифікуючу його інформацію. Ніяка сесійна інформація, включаючи cookies, не зберігається;

– у будь-який час можна відключити функцію «Захист від шахрайства» у меню «Налаштування – Розширені (Crtl-F12) – Безпека».

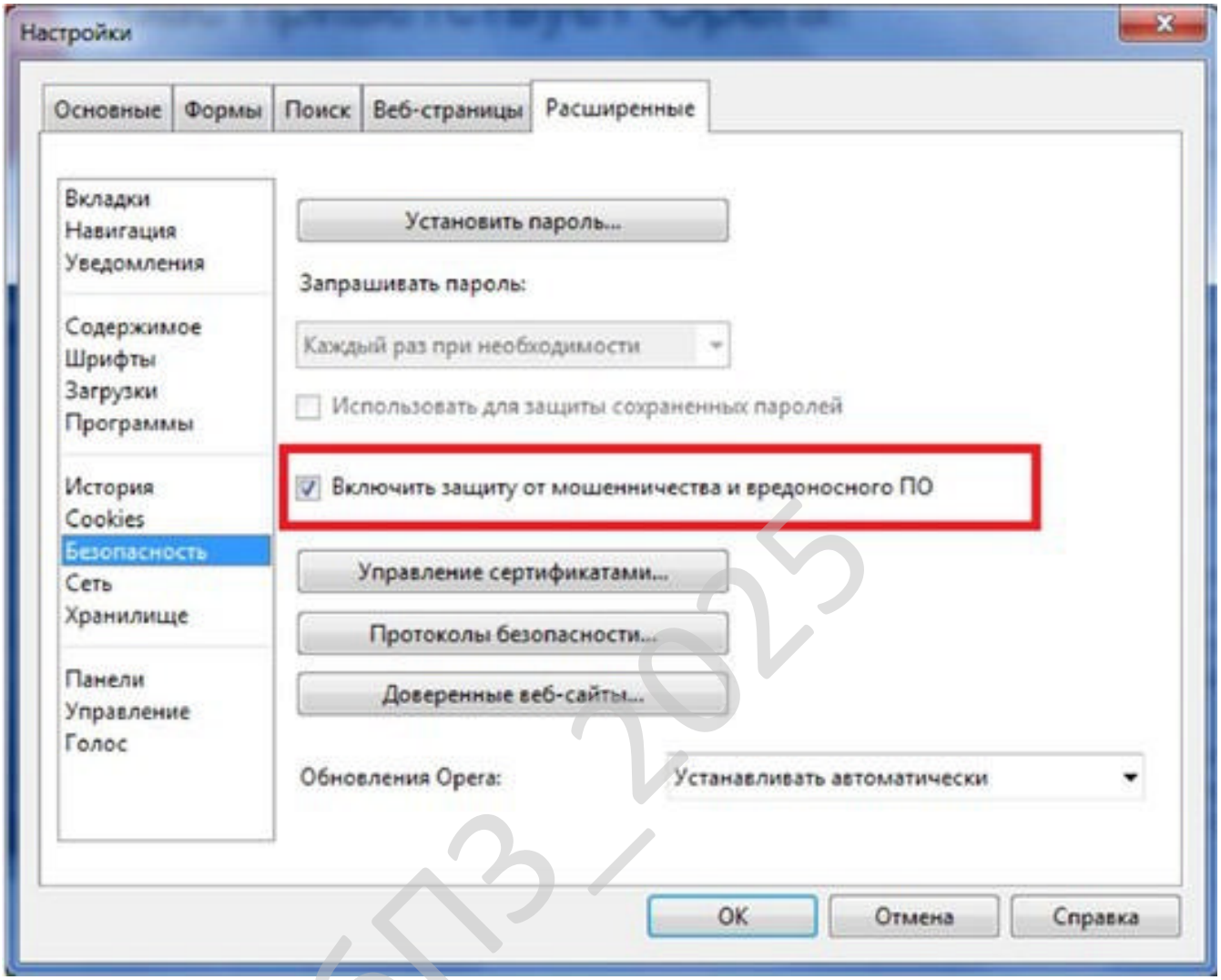


Рисунок 2.1 – Антифішинговий фільтр в Opera

Якщо веб-сайт знайдений у чорному списку, у браузері відкриється сторінка з попередженням. Користувачеві прийде вирішити, відвідувати цю підозрілу сторінку або повернутися на свою домашню. Механізм захисту від шахрайства не робить ніякого впливу на швидкість відкриття веб-сторінок.

Захист від фішингу й шкідливого ПЗ в Google Chrome

Функція безпечного перегляду Google Chrome, відповідальна за виявлення фішингу й шкідливого ПЗ, включена за замовчуванням. При спробі відвідування

сайту, підозрюваного у фішингу або поширенні шкідливого ПЗ, браузер показує попередження.

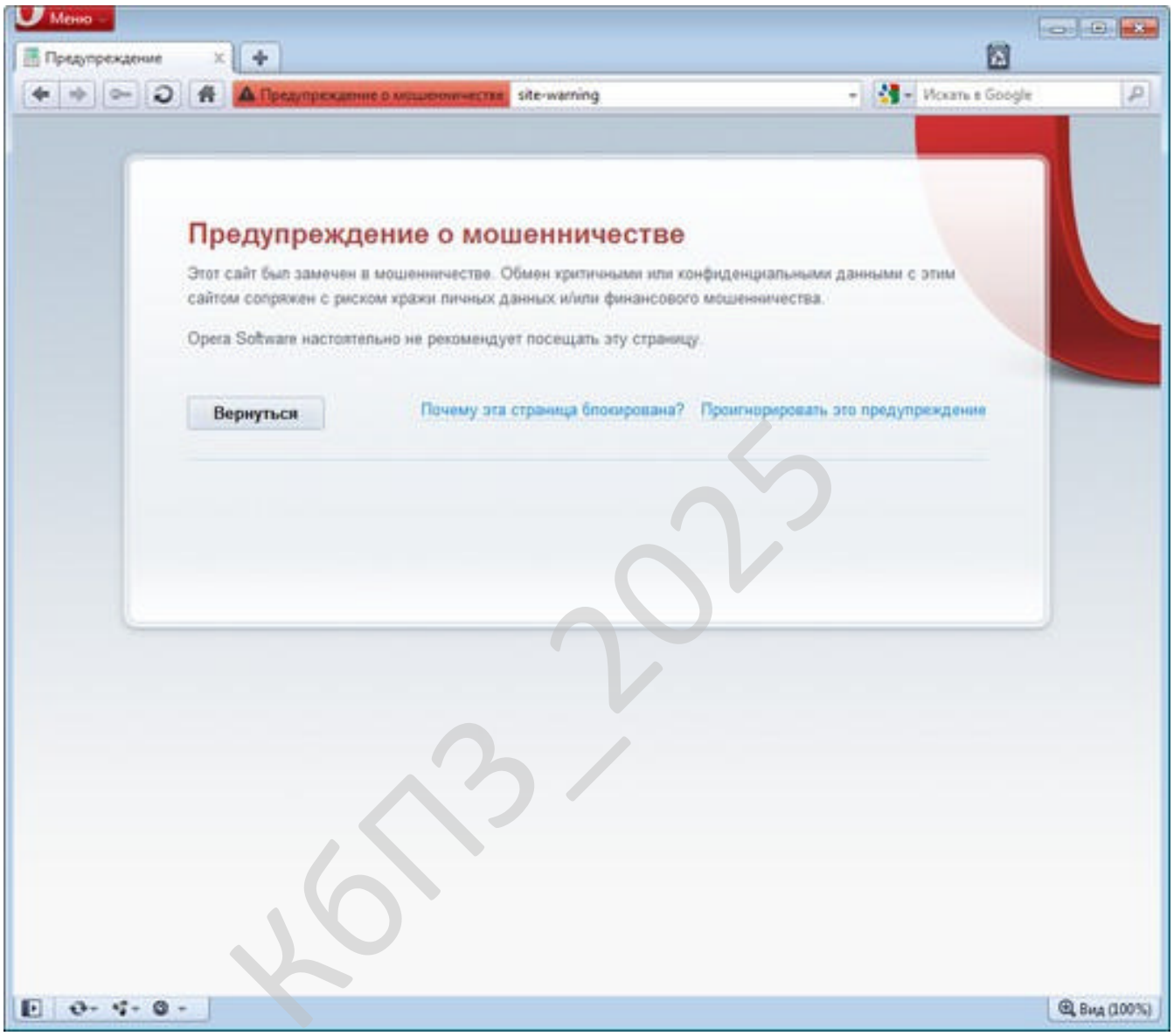


Рисунок 2.2 – Попередження про шахрайство в Opera

Принцип її роботи полягає в наступному. У браузер завантажується список з інформацією про сайти, які можуть містити шкідливе ПЗ або підозрюються у фішингу. Цей список не містить повні адреси URL кожного підозрілого сайту. Замість цього кожний URL хешується (змінюється таким

чином, що його не можна прочитати) і розділяється на фрагменти. Тільки частина кожного хешуємого URL включається в список у браузері.

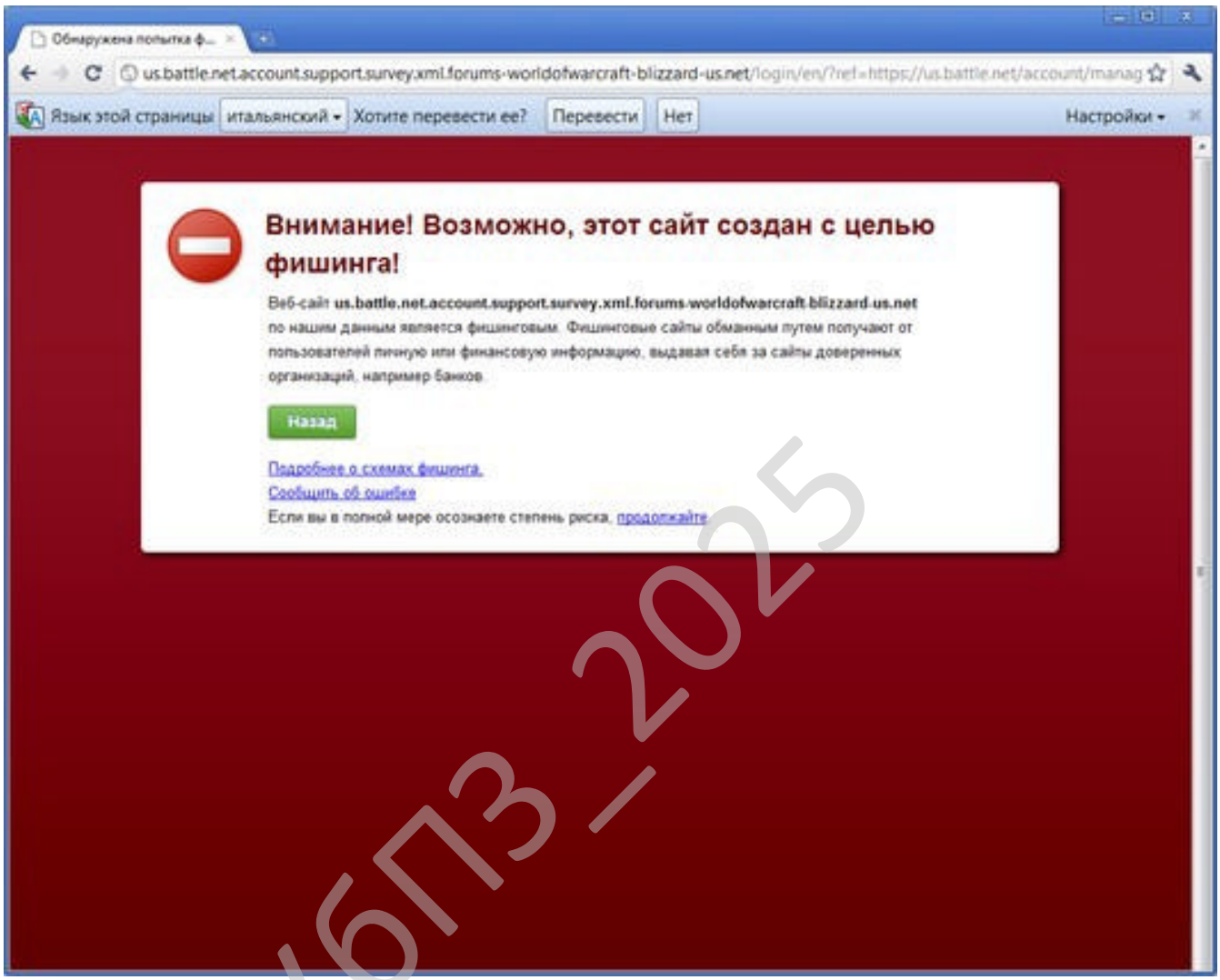


Рисунок 2.3 – Попередження про фішинговом сайт в Google Chrome

При роботі в Інтернеті браузер створює хешовані версії відвідуваних URL і перевіряє їх у відповідності зі списком. Якщо адреса відвідуваного сайту відповідає хешованому фрагменту URL у списку, браузер зв'яжеться із серверами Google і запросить повний список (а не тільки фрагменти) хешованих URL підозрілих сторінок. Потім комп'ютер визначить, чи є сайт підозрілим, і виведе відповідне попередження.

Відключити цю функцію можна в меню «Параметри – Розширені – Конфіденційність». Для цього досить зняти прапорець «Включити захист від фішингу й шкідливого ПЗ».

Захист від фішингу в Firefox

Технологія Phishing and Malware Protection перевіряє по базі фішингових і шкідливих сайтів сторінки, на які збирається перейти користувач. База зберігається на комп'ютері користувача й обновляється кожні 30 хв, якщо, звичайно, включений сам фільтр Phishing and Malware Protection.

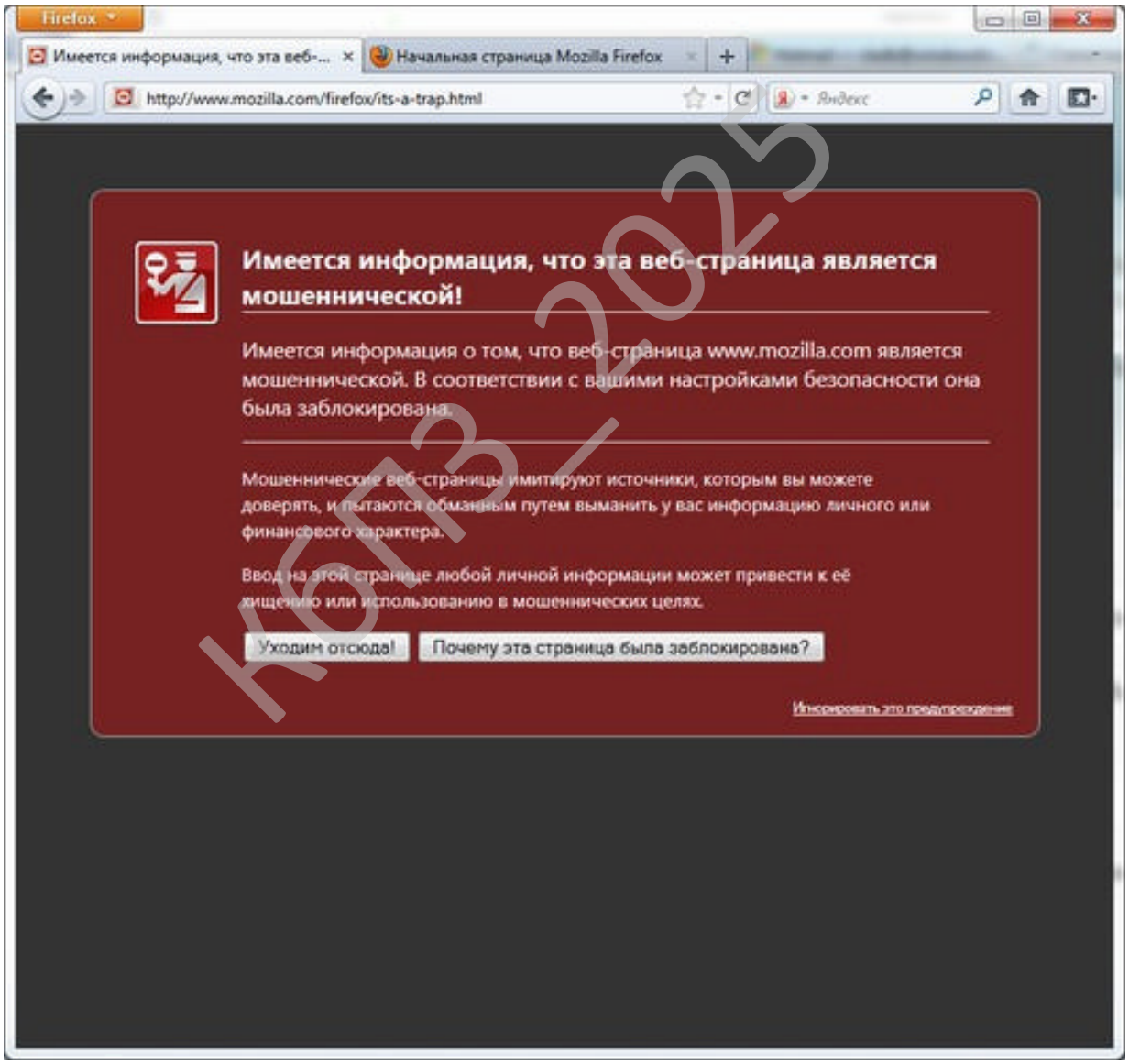


Рисунок 2.4 – Попередження Firefox про перехід на сайт, що містить шкідливе ПЗ

А якщо виявлено збіг адреси зі списком фішингових сайтів або сайтів, що містять шкідливе ПЗ, Firefox звертається до серверів компаній-партнерів, що надають такі чорні списки, і перед блокуванням сторінки ще раз перевіряє ще раз, чи не був вилучений сайт із бази після останнього її відновлення.

Фільтр Phishing and Malware Protection включений за замовчуванням.

Захист від фішингу й шкідливого ПЗ в Safari

За замовчуванням модуль захисту від фішингу в цьому браузері включений. Для пошуку фішингових сайтів він використовує технології Google.

Як тільки користувач намагається відкрити підозрілу сторінку в Safari, браузер з'єднується з Google і запитує інформацію із двох основних баз Google: бази фішингових посилань і бази посилань шкідливого ПЗ. При наявності збігу користувач повинен побачити сторінку з попередженням.

Фільтр SmartScreen в Internet Explorer

Починаючи з Internet Explorer 8 до складу IE входить фільтр SmartScreen – набір технологій, призначений для захисту користувачів від можливих інтернет-погроз, у тому числі погроз соціальної інженерії. Базується SmartScreen на технології фішингового фільтра й призначений для захисту користувачів від відомих шкідливих веб-вузлів. Крім того, даний фільтр включає захист від ClickJacking, технології, застосовуваної для перехоплення клавіш, перекручування веб-сторінок і т.буд. За замовчуванням він включений.

Фільтр SmartScreen в Internet Explorer використовує відразу кілька технологій. У першу чергу відбувається порівняння адреси відвідуваного сайту зі списком відомих шахрайських і шкідливих сайтів. Якщо сайт знайдений у цьому списку, більше перевірок не виробляється. У противному випадку він аналізується на предмет наявності ознак, характерних для шахрайських сайтів. Також можливе відправлення адреси того сайту, куди користувач збирається зайти, онлайн-службі Microsoft, що шукає його в списку фішингових і шкідливих сайтів. Причому доступ до онлайн-служби виробляється асинхронно по SSL-з'єднанню, так що це не позначається на швидкості завантаження сторінок. Однак звертання до даної служби користувач може заборонити.

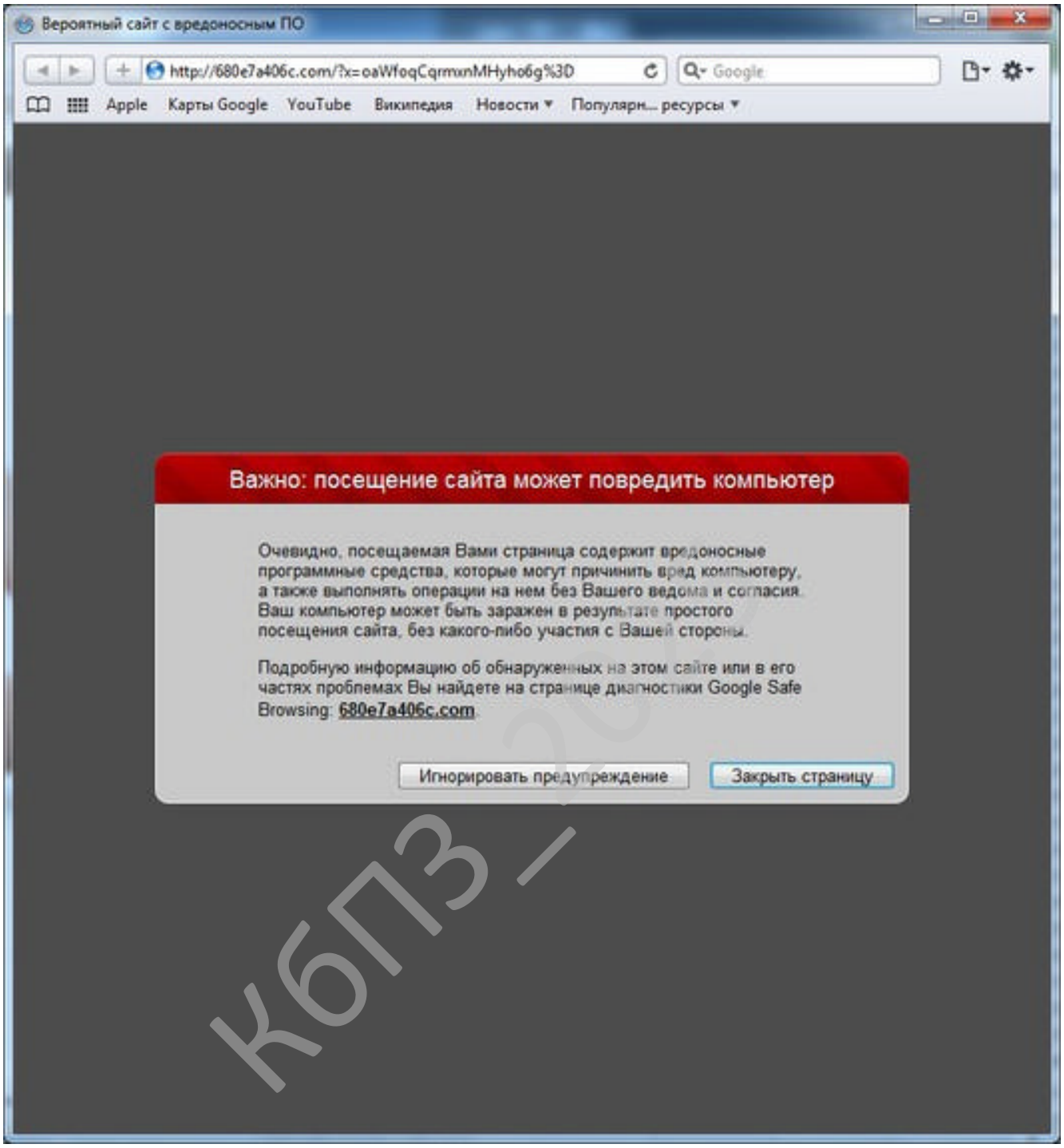


Рисунок 2.5 – Попередження про перехід на сайт, що містить шкідливе ПЗ, у браузері Safari

Щоб зменшити мережний трафік, на клієнтському комп'ютері зберігається зашифрований DAT-файл зі списком тисяч найбільш відвідуваних вузлів; всі включені в цей список не піддаються перевірці фільтром SmartScreen.

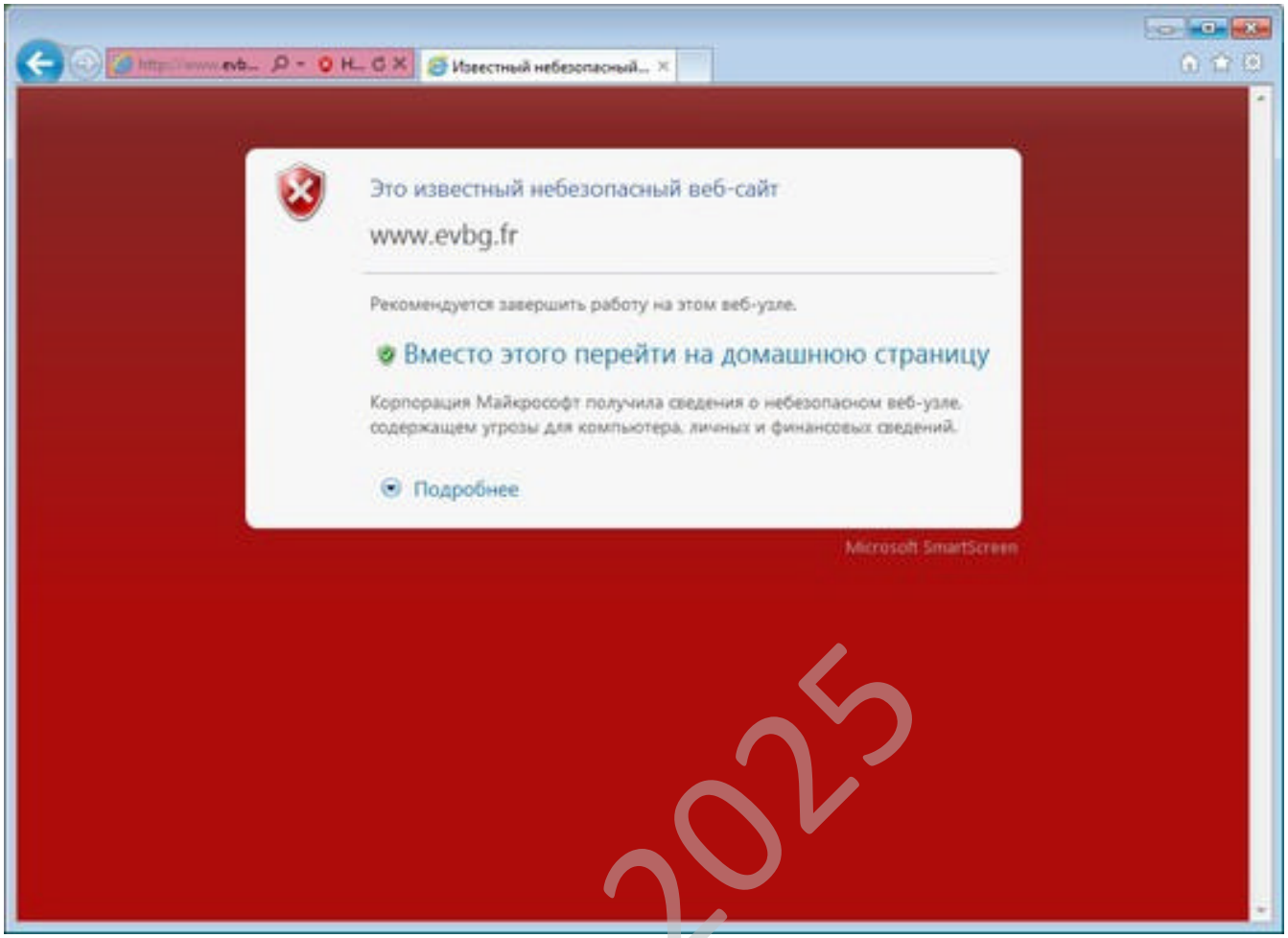


Рисунок 2.6 – Попередження, видаване фільтром SmartScreen в IE9 при спробі зайти на небезпечний сайт

Для захисту від фішингу й експлойтів фільтр SmartScreen досліджує рядок URL цілком, а не підмножина адрес URL, на які заходив користувач, а виходить, службі URS можуть бути передані особисті відомості, оскільки іноді вони перебувають у самому рядку URL.

2.2 Обґрунтування вибору засобів для побудови системи кібербезпеки та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність

розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій. Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети). Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на

Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net. Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи. Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень. Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх. Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий. Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Банківський фішинг

Фішингові сайти традиційно приносять шахраям стабільний заробіток і вселяють почуття безкарності. Не кожний користувач, потрапивши на сторінку для введення ім'я й пароля, помітить невеликі невідповідності квітів або шрифтів у порівнянні з оригінальною сторінкою. На це й розраховують кіберзлочинці, незмінно вдосконалюючи шахрайські технології й придумуючи нові виверти.

Фахівці лабораторії безпеки G Data SecurityLabs проаналізували фішингові сторінки, виявлені в 2014 – початку 2015 по усьому світі, і визначили, яку тематику шахраї використовують для обманів найчастіше.

Фішинг звичайно здійснюється за наступною схемою: шахраї створюють стартову веб-сторінку банку, гри або соціальної мережі, куди користувач повинен внести свої дані доступу. За допомогою свого сервера вони ставлять фальшивку онлайн, а його веб-адреса найчастіше настільки схожа на оригінальну, що недосвідчений користувач навіть не помітить підміни. Користувач відкриває підроблену сторінку, уводить свої дані, після чого шахрай використовує їх на свій розсуд.

За підсумками дослідження, перше місце з результатом 63 % зайняли фішингові сайти, присвячені тематиці банків і електронної комерції. Підроблені сторінки для входу в особистий кабінет від PayPal були особливо популярні серед фішерів. Створюючи подібні сайти, шахраї прагнуть викрасти персональних даних власників цих аккаунтів для того, щоб украсти їхні гроші або використовувати для іншого роду шахрайства. До речі, виверту банківського фішингу виявилися й самими дохідними. Саме тому дана тематика залишається на чолі рейтингу.

З однаковими показниками в 13 % на другому й третім місці виявилися фішингові сайти, присвячені темі соціальних мереж, чатів і e-mail, а також іграм. Дані доступу на подібні сервіси непогано продаються на підпільному ринку персональних даних, наприклад, вартість аккаунтів для таких онлайн-ігор, як World of Warcraft залежить від рівня й «оснащення» гравця. Також свою ціну має зброя й віртуальна валюта. Дні, коли гра робила непорівнянну приємність від пройдених рівнів, давно пройшли. Зараз вона перетворилася засіб заробляння грошей для несумлінних гравців.

Якщо говорити про соціальні мережі, то крім продажу даних доступу на користувальницький профіль шахраї зацікавлені в розсиланні спаму й шкідливого коду із чужих аккаунтів. Користувач, що одержав повідомлення від свого «віртуального» друга більш охоче перейде по посиланню, чим, якби повідомлення прийшло від незнайомої людини.

Із часток усього 10 % четверте місце в рейтингу фішингових сайтів займає тема онлайн-магазинів і аукціонів. Ця категорія включає, наприклад, фішингові сайти для покупців магазина eBay. Якщо фішери одержать дані доступу до аккаунту, вони можуть перемінити банківські реквізити на вхідні платежі для оплати покупок на їх особистий банківський рахунок. Так вони можуть одержати гроші, переведені за покупки. Це особливо небезпечно для комерційних провайдерів, які продають сотні товарів у день.

SMS-шахрайство

Мобільні оператори зараз фактично є системами для організації мікроплатежів. Тому всі частіше шахраї використовують телефон для того, щоб залізти в кишеню користувачів мобільного зв'язку. Способи для цього можуть бути найрізноманітнішими: від розсилання SMS-спаму до зараження троянцем смартфона або вимоги викупу за розблокування домашнього комп'ютера. У цьому розділі ми розглянемо найбільш популярні способи шахрайства за допомогою мобільних телефонів.

Пік SMS-шахрайства по завіреннях мобільних операторів довівся на жовтень минулого року, коли в українському сегменті мережі Інтернет вибухнула епідемія троянської програми Winlock. Вона вимагала для розблокування комп'ютера послати SMS на спеціальний платний номер, і користувачів, які скористалися подібною платною послугою виявилося досить багато. Тому основні оператори мобільних платежів зібралися й у листопаді минулого року домовилися про спільне рішення проблем подібного роду. Зокрема, були вироблені загальні правила підключення контент-провайдерів і налагоджений контакт між службами захисту від шахраїв різних платіжних систем. Зараз деякі оператори сформували розділи на своїх сайтах, які присвячені шахрайству за допомогою SMS.

У шахраїв є кілька способів одержання грошей з рахунків мобільних операторів: платні сервіси, відправлення на які SMS досить дорога, завантаження шкідливого контенту, прямий переклад грошей на телефон, виманювання коду карти оплати й SMS-Реклама. Розберемо кожний із цих способів докладніше.

Платні сервіси

Деякі короткі номери мають таку неприємну властивість – при посилці SMS на них з рахунку мобільного телефону списується певна кількість засобів на користь оператора. При цьому ціни можуть доходити до 100 грн. за одне повідомлення. Хоча шахрай при цьому одержував тільки свій відсоток, що завжди менше половини списаної в клієнта суми, проте, шахраї на великій операторській базі можуть заробити пристойні гроші. До цього ж типу атак можна віднести дзвінки на платні номери телефонів.

Правда, щоб одержувати гроші цим способом шахрай повинен укласти контракт із платіжною системою мобільних платежів. Їх в Україні 7 – 8 великих, тому шахрай, готуючись до своєї афери, раніше міг перебрати кілька подібних операторів, поки не знаходив того, хто погодиться переводити йому гроші. Правда після випадку з Winlock і домовленості між платіжними системами сформована загальна база підозрілих проектів, і такий спосіб одержання грошей

став більше проблематичним. Крім того, шахрая можна було обчислити за номером телефону й контракту із платіжною системою.

Проте, цей тип одержання грошей в Україні дуже популярний серед шахраїв. Зокрема він використовується в наступних атаках:

– лотереї: шахраї пропонують щось розіграти, для чого досить послати SMS на зазначений номер;

– фальшиві листівки: це тип спаму, коли користувачеві повідомляється про одержання листівки, для чого він повинен перейти по посиланню, однак самої листівки одержати не вдається, хоча гроші за доступ до неї все-таки списуються;

– лже-робота: тип спаму, у якому пропонується високооплачувана робота, але для її одержання потрібно відправити SMS або передзвонити на платний номер;

– життя без спаму: на телефонні номери, які засвічені в спамерских базах, може прийти пропозиція відписатися від розсилання спаму за допомогою "безкоштовного" SMS;

– трояни й фальшиві антивіруси: на комп'ютері запускається троянец, що вимагає перерахування грошей за допомогою SMS;

– мобільні трояни: деякі вредоноси для мобільних телефонів розсилають SMS або виконують дзвінки на "дорогі" номери;

– нав'язана підписка: тип спаму, у якому пропонують скачати безкоштовний контент або одержати доступ до певних послуг, однак у посиланні "зашитий" код активації SMS-підписки, за користування якої з рахунку користувача регулярно будуть зніматися певні гроші;

– телевікторини: деякі телевізійні передачі націлені на викачування грошей з мобільного телефону.

Для захисту від цього виду шахрайства рекомендується спочатку довідатися точну вартість одного SMS на зазначений номер, а потім уже приймати рішення про їхнє відправлення. Списки "дорогих" номерів можна довідатися, звернувшись у службу підтримки свого мобільного оператора або платіжної системи.

Передоплачені карти й поповнення рахунку

Гроші від користувачів мобільного зв'язку можна одержати у вигляді коду передоплаченої карти або оплати через термінал. Схема цього шахрайства проста заснована на тому, щоб змусити людину покласти гроші на чужий рахунок.

Фіктивна допомога: жертві приходить приблизно таке повідомлення "Потрапив в аварію. Пишу із чужого телефону. Переведи на нього гроші". Перш, ніж допомагати в такій ситуації рекомендується передзвонити людині, від імені якого представляється шахрай.

SMS-прохання: це варіант попереднього шахрайства, але розсилається приблизно таке повідомлення: "У мене проблеми, подзвони по такому-те номері. Якщо номер не відповідає, поклади на нього гроші й передзвони".

"Акції" операторів: шахраї надсилають SMS нібито від імені оператора зв'язку з повідомленням про проведену акцію, наприклад, з безкоштовними дзвінками по країні. Для участі в акції пропонується в певний строк переслати на номери "техпідтримки" оператора (номер додаються) коди передоплачених карт.

Фальшива лотерея: дзвінок нібито з радіо або телебачення з повідомленням, що ваш номер виграв у лотерею приз – мобільний телефон. Причому його відразу підключать, для чого потрібно купити передоплачену карту й повідомити код діджею. Оскільки радіо проводить лотереї, як правило, у прямому ефірі, те перш ніж диктувати код рекомендується спочатку послухати зазначену радіостанцію.

Прямий переклад грошей

Деякі оператори мобільного зв'язку реалізували систему прямого перекладу грошей з одного номера на інший. Це дозволяє реалізувати нові форми шахрайства, де різними способами користувачів такого оператора змушують набрати код перекладу грошей на чужий рахунок. Найбільш відомі шахрайства цього типу наступні:

Техобслуговування: дзвінок від нібито співробітника мобільного оператора, що просить набрати на клавіатурі комбінацію певних клавіш. Ця комбінація може бути, у тому числі, і зверненням до служби прямого перекладу коштів.

Помилковий переклад: дзвінок з незнайомого номера з повідомленням про те, що помилково на ваш номер перевели певну суму. Закінчується така розмова проханням переказати гроші назад, хоча цілком можливо, що ні хто ні яких грошей і не переводив.

Терміновий дзвінок: на вулиці до вас звертаються, щоб зробити терміновий дзвінок. Коли телефон попадає в руки шахрая, він переказує гроші на свій рахунок або дзвонить на "свою" платну службу. Рекомендується набирати номер самостійно й стежити, щоб він не починався із символу "*".

Мобільний спам

Властиво одержуваний SMS-спам не вимагає витрати від користувача – за нього платить рекламодавець, проте, видаляти сміттєві SMS неприємно. Тому збір телефонних адрес у базу даних спамера також можна віднести до мобільних погроз. Перш, ніж продати базу перевірених адрес покупцеві шахрай повинен її веріфікувати, щоб мінімізувати витрати на що відсилаються SMS. Для верифікації можуть бути використані самі безневинні питання, типу "Як справи?", "Чим займаєшся" або просто "Превед!", однак відповідати на подібні питання отримані з незнайомих номерів, не рекомендується.

Властиво, є ще багато найрізноманітніших погроз, які прив'язані до конкретних сервісів. Наприклад, якщо власник мобільного телефону користується ще й послугами мобільного банкінгу, те в нього за допомогою різних технік соціальної інженерії можуть виманювати реквізити цього сервісу. Втім, поки в Україні подібні сервіси досить рідкі, тому й атакуються вони рідко. Можливо, у міру розвитку різних сервісів шахраї придумають різні способи одержання грошей від їхнього використання.

З наведених методів шахрайства видно, що чисто технічні методи атаки використовуються досить рідко – найбільше атак побудовано по принципах соціальної інженерії. Захиститися від них просто – потрібно знати, що як саме нападають розраховують одержати з вас гроші, і не йти в них на приводу. Для цього варто ознайомитися з рекомендаціями самих операторів зв'язку, які завели відповідні розділи на своїх сайтах.

Longlining

Proofpoint, американський провайдер хмарних захисних рішень, попереджає про появу різновиду фішинг-атак, які успішно обходять традиційні репутаційні фільтри й системи сигнатурного аналізу. За півроку експерти Proofpoint зареєстрували десятки атак нового типу по усьому світі. У компанії їх називають «longlining» – атаки по типі ярусного лову (застосовуються при промисловому видобутку риби коштовних порід). Ці фішинг-атаки ефективно сполучать швидкість і масовість розсилань із високою варіативністю контенту, що значно утрудняє їхнє виявлення наявними засобами.

Proofpoint удалося задокументувати ряд вторгнень нового типу. Одна із хвиль у жовтні минулого року пройшла й в Україні. У ході атаки всього за 3 години було розіслано 135 тис. повідомлень у більш ніж 80 компаній. Зловмисники використовували понад 28 тис. IP-адрес й більше 35 тис. імен відправників. Посилання усередині повідомлень вели на десятки скомпрометованих легальних веб-сайтів, куди були попередньо впроваджені джерела drive-by завантажень і асоційованих з фішинговими URL. Через різноманітний контент, відправників і іншого в жодній із цільових компаній не було виявлено більше 3-х повідомлень із однаковими характеристиками, тому ідентифікувати вторгнення як цільові атаки не вдалося. По даним Proofpoint, на частку даного розсилання в цільових організаціях довелося менш 0,06% сукупного поштового трафіку (тоді як спам-реклама склала 19%, листа зі шкідливим вкладенням – 11%).

Експерти відзначають, що при такій масовості longline-розсилань їхня ефективність викликає більшу тривогу – понад 10% шкідливих URL не тільки пройшли всі корпоративні фільтри, але й були активовані. А кожний п'ятий фатальний «клік» (19%) був зроблений, коли службовець перевіряв корпоративну пошту, перебуваючи за межами захищеного простору – з будинку, у дорозі або з мобільного пристрою.

Фішинг в в Dota 2

У гравців в Dota 2 виманюють особисті дані, обіцяючи віртуальні багатства. Гравцям Dota 2 варто уважно поставитися до одержуваних листів і посилань. Згідно даним Malwarebytes, зараз по Мережі гуляє нова шахрайська сторінка. Фішери пропонують унікальні віртуальні предмети в обмін на особисту інформацію.

Підозрілий сайт називається steamonlinereward.webs.com. На його сторінках користувачам пропонуються рідкі ключі й Arcana-предмети для гри Dota 2 – однієї із самих популярних у сервісі Steam (аудиторія оцінюється в 700 тисяч активних геймерів). Користувачів просять увести email із сервісу Yahoo, а також указати пароль, щоб одержати обіцяні речі. Після того як дані були уведені, сторінка видає спливаюче вікно з повідомленням «повернемося до вас, як тільки це буде можливо».

Кілька днів назад про цей сайт написали на форумах Steam. Всім користувачам Dota 2 Valve радить більш уважно ставитися до облікових записів в Steam і триматися подалі від будь-яких сайтів, що обіцяють сумнівні внутриігрові предмети.

Фішингові й шкідливі листи

Відомі компанії й бренди є улюбленими мішенями шахраїв. Адже привернути увагу користувачів до листів, написаним нібито популярними й пізнаваними компаніями, набагато простіше, а виходить, і шанси, що користувач стане жертвою шахраїв, теж високі.

У цьому розділі ми докладно вивчимо фішингові й шкідливі листи, які розсилаються зловмисниками від імені міжнародних служб доставки. Найбільш популярними у фішерів є німецька компанія DHL, американські компанії FedEx, United Parcel Service (UPS) і голландська компанія TNT. Всі ці компанії є міжнародними, мають філії в найбільших країнах миру й мільйони клієнтів. І оскільки перераховані компанії надають послуги в одній сфері спамери, використовуючи їхні імена, прибігають до одних і тих же методів і прийомів для створення шахрайських листів і обману користувачів.

Фішери переслідують наступні цілі:

– Крадіжка конфіденційних даних користувачів (даних банківських карт, логінів і паролів від особистих кабінетів) в основному за допомогою підроблених веб-сторінок, що імітують офіційні сторінки сайту. Головна риса фішингу – користувач сам передає шахраям свої персональні дані, заповнюючи поля на підроблених сторінках або відправляючи їх поштою.

– Установка на комп'ютер користувача різних шкідливих програм, які використовуються не тільки для відстеження дій користувачів і крадіжки різної персональної інформації, але й для організації ботнетів для розсилання спаму й проведення DDoS-атак.

Заголовки шахрайських листів

Поле відправника (From)

Структурно адреса в поле From виглядає як: Ім'я відправника. Щоб увести одержувача в оману, шахраї можуть міняти будь-який елемент адреси, у тому числі роблячи його схожим на офіційну адресу служби доставки.

У шахрайських листах можна виділити кілька груп електронних адрес:

– Електронні адреси, схожі на легітимні загальні адреси, що належать компаніям. У них як ім'я відправника звичайно використовується безпосередньо назва компанії (DHL INC, TNT COURIER SERVICE, Fedex і т.д.). Як ім'я поштової скриньки часто використовуються слова info, service, noreply, mail, support, які характерні для адрес електронної пошти, що використовуються для розсилання різних офіційних повідомлень. Як доменне ім'я сервера вказуються реальні або дуже правдоподібні домени компаній.

– Адреси, не схожі на легітимні адреси компанії. У таких адресах як ім'я відправника також використовується назва компанії (FedEx, DHL Service, FedEx.com), а от доменне ім'я звичайно належить різним сервісам безкоштовної пошти або зовсім іншим компаніям. Як адресу електронної пошти спамери можуть використовувати адреси реальних користувачів (зламані електронні адреси, адреси, зібрані з різних відкритих джерел) або згенеровані автоматично адреси. Останні звичайно являють собою довільну послідовність букв, слів і цифр.

– Адреси, схожі на електронні адреси співробітників компаній. Як ім'я відправника в такій адресі може стояти ім'я й прізвище, а також назва компанії або посада (Courier, Manager і т.д.). В імені поштової скриньки звичайно вказуються ті ж ім'я й прізвище, що й в імені відправника, тому що різні дані можуть насторожити одержувача шахрайського листа. Як доменне ім'я може використовуватися як справжній домен компанії, так і інші домени, не пов'язані з компаніями служб доставки.

– Адреси, у яких вказується тільки адреса відправника без вказівки його імені.

При вивченні адреси відправника не варто забувати, що для того щоб у полі відправника стояв реальний домен компанії, шахраям зовсім не обов'язково зламувати сервери компанії, досить підставити на адресу поля From потрібне доменне ім'я сервера.

Поле Subject

Тема шахрайського листа повинна привертати увагу одержувача й спонукувати його відкрити лист, але в той же час бути правдоподібною. Тому спамери вибирають як тема загальні фрази, характерні для офіційних листів служб доставки. Відправляючи посилку або документи, клієнти в більшому або меншому ступені турбуються за успішну доставку й, звичайно, намагаються не тільки активно відслідковувати її пересування, але й читати всі інформаційні листи від служб доставки.

Найбільш популярними темами є:

– Теми, пов'язані з доставкою/відправленням відправлень (повідомлення про відправлення, статус доставки, підтвердження відправлення, документи про відправлення, інформація про доставку й т.д.).

– Теми, пов'язані з відстеженням відправлень, інформацією про замовлення й рахунок на оплату (номер відправлення, відстеження відправлення й т.д.).

– Теми, пов'язані з повідомленнями про повідомлення й аккаунтах (створення й підтвердження аккаунта, одержання нових повідомлень і т.д.).

Оформлення листа

Особливу увагу шахраї приділяють оформленню електронного листа. Їхня головна мета – змусити одержувача повірити в правдивість листа. Адже якщо воно здасться підозрілим, те потенційна жертва, швидше за все, просто видалить його, незважаючи на приваблюючу увагу тему й гадане сьогодення адреса відправника. Розглянемо основні прийоми, які шахраї використовують для додання листам легітимного виду.

Графічне оформлення

Великі компанії, що успішно працюють на світовому ринку, неодмінно мають свій власний фірмовий стиль, що включає словесний товарний знак, графічний товарний знак, фірмовий шрифт, слоган, колірну гаму оформлення офіційного сайту, розсилань і рекламних роликів і інші компоненти. Деякі з перерахованих елементів використовуються зловмисниками в оформленні шахрайських листів з метою привернути увагу одержувача й додати листу справжній вид. Зокрема, частіше інших фішери використовують логотип, оскільки цей елемент є унікальним для будь-якої компанії, по ньому довідаються й виділяють компанію серед інших організацій.

Текстове оформлення

У тексті більшості офіційних листів можна зустріти ряд стандартних фраз, особливо якщо мова йде про типові повідомлення, згенерованому і відправленому автоматично. Також у таких листах часто присутні контакти й посилання на офіційні ресурси компанії-відправника. Тому щоб зробити текст фальшивого листа схожим на справжнє повідомлення від служб доставки шахраї використовують:

– Типові фрази, характерні для офіційних розсилань: Please do not reply to this email, This is automatically generated email, please do not reply, All rights reserved, Diese Versendung ist automatisch, Bitte beantworten Sie diese nicht, This communication contains proprietary information and may be confidential. Questo e' un email automatico, Si prega di non rispondere і ін.

– Посилання на офіційні сторінки компанії. Не всі посилання, зазначені в шахрайському листі, є фішинговими – спамери також можуть використовувати справжні посилання, що ведуть на офіційні ресурси, для додання своєму листу легітимного виду й обходу засобів фільтрації спаму.

– Контакти для зворотного зв'язка. Шахраї нерідко вказують дані (ім'я, прізвище, посада, адреса офісу) конкретного співробітника – відправника листа або загальних контактів компанії. Відзначимо, що такими контактами можуть бути й справжні, і вигадані адреси й телефони.

Зміст листа

Для шахраїв, що розсилають фальшиві листи, важливо не тільки переконати одержувача в правдивості листа, але й змусити передбачувану жертву добровільно виконати потрібні дії, наприклад надати персональну інформацію або встановити шкідливий файл. Для цього зловмисники використовують прийоми психологічного впливу на одержувача. У цьому випадку головним інструментом у руках фішерів є текст листа, його зміст.

У шахрайських повідомленнях, присланих нібито від служб доставки, зловмисники часто використовують наступні прийоми:

– Повідомлення про різні помилки й причини їхнього виникнення (невдала доставка, відсутність інформації, неправильна адреса, відсутність одержувача за адресою доставки). Звичайно такі фрази пов'язані з доставкою відправлення, тому що саме в цій сфері послуг працюють розглянуті нами компанії. Тому повідомлення про помилку в доставці від логістичної компанії не викликає в одержувача ніяких підозр, особливо якщо в листі втримується пояснення причин недоставляння і яких-небудь подробиць.

– Прохання виконати що-небудь зі згадуванням можливих санкцій у випадку невиконання. Наприклад, "заберіть відправлення протягом 5 днів, інакше воно буде повернуто відправникові". Такі фрази шахраї використовують, щоб змусити одержувача негайно виконати те, що написано в листі. У цьому випадку фішери розраховують, що користувач, боячись не одержати посилку або

виявитися змушеним платити пені, не стане довго роздумувати й під впливом емоцій сам передасть персональні дані або запустить шкідливий файл.

– Фрази із вказівкою, що перебуває у вкладенні або по посиланню (рахунок, докладна інформація, документи). Якщо користувач не знає або не впевнений у тім, що перебуває по посиланню або у вкладенні, він навряд чи буде діяти так, як потрібно зловмисникам. Тому шахраї видають підроблені сторінки за сторінки офіційних сайтів, а зловреди в архіві – за документи, що містять різну інформацію про відправлення. Крім того, якщо в тексті повідомлення говориться, що у вкладенні, наприклад, перебуває транспортна накладна, те й сам шкідливий архів буде мати співзвучна назва, наприклад, "транспортна накладна.zip". Це відноситься й до фішингових посилань – зловмисники прив'язують їх до відповідних фраз у тексті, наприклад, "інформація про доставку". Цей простий прийом використовується спамерами для того, щоб в одержувача не виникло сумнівів у тім, що у вкладенні або по посиланню перебуває саме те, про що говориться в тексті повідомлення.

– Фрази про необхідність виконати які-небудь дії (перейти по посиланню, відкрити вкладення, роздрукувати файл і т.д.).

Представимо, що шахраям все-таки вдалося переконати одержувача в правдивості листа. Далі необхідно повідомити потенційній жертві, що варто зробити для рішення описаної в листі проблеми, тому що саме виконання зазначених у тілі листа дій і є кінцевою метою зловмисників. Отут для шахраїв важливо не просто повідомити одержувача, що від нього потрібно зробити, а повідомити так, щоб він правильно зрозумів написане в листі. Щоб уникнути непорозуміння з боку одержувача, у тексті листа часто зустрічаються вказівки конкретних дій, які користувач повинен виконати.

Що може мінятися в тексті

Обман користувача не єдине завдання зловмисників. Спочатку їм треба обійти спам-фільтри й доставити листа в електронні ящики потенційних жертв. Одним з найбільш популярних і давно використовуваних методів обходу спам-

фільтрів є зміна текстових фрагментів листа. У сучасних програмах розсилання спам-повідомлень є широкі можливості по генерації численних варіантів змін у тексті. Мінливий від листа до листа текст робить лист унікальним, а вказівка різної персональної інформації в листах одного розсилання, наприклад номера відправлення, звернення, дати, допомагає переконати одержувача, що лист адресований саме йому. Крім того, шахраї можуть розсилати листа, оформлені по одному шаблоні, протягом декількох місяців, для цього їм необхідно лише міняти деякі елементи в тексті.

У шахрайських повідомленнях від служб доставки міняються:

– Інформація про замовлення/відправлення. Сюди в основному відноситься номер відправлення, номер відстеження відправлення, строки доставки відправлення й т.д.

– Контактні дані й імена відправників, назви компаній. У деяких розсиланнях для зворотного зв'язка із представником компанії, що відправили лист, вказуються телефони або електронні адреси. Саме ці дані й змінюються від листа до листа. Крім того, імена представників компанії й безпосередньо назви компаній також можуть мінятися.

– Ім'я вкладення. В основному це відноситься до шкідливих вкладень, назви яких міняються в листах одного розсилання, однак при цьому під різними назвами ховається та сама шкідлива програма.

– Посилання. У фішингових листах і листах зі шкідливими посиланнями спамери часто міняють саме адреси посилань, маскуючи їх, у тому числі за допомогою різних сервісів скорочення посилань. В основному такі посилання швидко блокуються сучасними антивірусами.

– Фрази із вказівкою кількісних характеристик і дати. У число кількісних характеристик входять тривалість (дні, годинники), кількість коштів (наприклад, пені) і дата (число й місяць).

– Звернення. Як звернення спамери звичайно підставляють у тіло листа адреса електронного ящика й/або ім'я одержувача. Іноді замість них можуть вказуватися загальні слова (client, customer і т.д.).

– Інші текстові фрагменти. Деякі слова заміняють інакше кажучи, близькими за значенням, не міняючи при цьому змісту пропозиції в цілому. Наприклад, слово package можна замінити словом parcel, а enclosed – словом attached.

Підроблені сторінки

Для крадіжки особистої інформації користувачів шахраї створюють фішингові HTML-сторінки, оформлення яких частково або повністю копіює офіційний сайт якої-небудь компанії. Якщо жертва шахраїв уводить на такій сторінці персональну інформацію (банківські дані, логіни й паролі), остання негайно попадає в руки зловмисників.

Для маскування посилань, що ведуть на фішингові сторінки, шахраї часто використовують популярні безкоштовні сервіси скорочення посилань. Крім того, більшість сервісів пропонують своїм клієнтам можливість переглядати статистику по розміщеному короткому посиланню, що дає шахраям додаткову інформацію, наприклад про кількість переходів по посиланню й т.д. Фішингові сторінки можуть бути розміщені на спеціально зареєстрованих доменах, які звичайно мають короткий строк життя, а також на зламаних доменах, чий власник може й не підозрювати про те, що його веб-сайт використовується в шахрайських цілях.

Розглянемо підроблений лист, відправлений від імені компанії FedEx, у якому одержувача просять обновити інформацію облікового запису на сайті. У тексті листа одержувач бачить посилання на сторінку офіційного сайту компанії, однак реальна адреса, на який перенаправляють користувача зовсім не схожий на легітимний і розташований на безкоштовному сервісі скорочення посилань. Це стає очевидно, якщо навести курсор на посилання.

Кликнувши по посиланню, користувач попадає на шахрайську сторінку, що імітує офіційний сайт компанії FedEx, де необхідно ввести логін і пароль для входу в особистий кабінет. Як тільки користувач заповнить поля й натисне кнопку "Login", уведена інформація передається шахраям, які одержують доступ

до особистого кабінету жертви. Слід зазначити, що часто вкладки меню й численні посилання на фішинговій сторінці неактивні, тобто при натисканні на передбачуване посилання переходу на потрібну сторінку не відбувається. Однак у деяких випадках фішери підробляють всі посилання на сторінці, щоб у користувача не виникло сумнівів у легітимності фішингової сторінки. Часте оформлення сторінки тільки схоже, а не копіює офіційну сторінку повністю. Якщо придивитися до деталей, то можна помітити деякі відмінності в оформленні справжньої й підробленої сторінок. Однак у більшості випадків користувач не обертає уваги на дрібні розходження, і шахраї успішно використовують це для крадіжки персональної інформації.

Розглянемо ще один приклад листа від імені компанії FedEx, цього разу зі шкідливим посиланням. У листі повідомляється, що через відсутність важливої інформації доставка не може бути здійснена. І тепер для перевірки особистості користувачеві необхідно пройти по зазначеному посиланню.

Посилання веде на шахрайську сторінку, де потенційній жертві пропонують скачати програму, що нібито повинна перевірити, чи є користувач одержувачем відправлення. Під видом програми, природно, ховається зловред – відома троянська програма Zeus, за допомогою якої шахраї одержують доступ не тільки до комп'ютера користувача, але й до всієї персональної інформації жертви.

Варто відзначити, що шахраї можуть не тільки вказувати адресу фішингового посилання в тілі листа, але й прикріплювати до листа вкладення у форматі HTML, що і є фішинговою сторінкою, призначеної для крадіжки персональної інформації користувачів. Для шахрайських розсилань від служб доставки використання HTML-вкладень у якості фішингових сторінок не характерно.

Шахрайські листи на різних мовах

Для розширення аудиторії одержувачів і замовників спамери освоюють всі нові мови. Крім традиційних англійської й німецької мов сьогодні в спам-трафіку можна зустріти листа на івриті, албанському й інших мовах, які кілька років

назад не зустрічалися в рекламних і шахрайських розсиланнях. Так, серед підроблених повідомлень від міжнародних служб доставки зустрічаються повідомлення, наприклад, на італійській і голландській мовах. Такі листи не мають ніяких особливостей у порівнянні з листами, написаними на англійській і німецькій мовах, – для обману користувачів зловмисники прибігають до тих же вивертам.

Наприклад, у фальшивому повідомленні про доставку від компанії FedEx, написаному італійською мовою, користувачеві необхідно було підтвердити свою особистість, пройшовши по шахрайському посиланню.

Лист ще з одного розсилання італійською мовою містило шкідливий архів, у якому перебувала троянська програма Zeus/Zbot, використовувана для крадіжки персональних даних. У шахрайському листі повідомлялося, що профіль користувача на сайті компанії був оновлений і більше детальна інформація про це перебуває в архіві.

В іншому підробленому повідомленні, написаному голландською мовою від імені компанії TNT, повідомлялося, що новий рахунок сформований і його оригінал перебуває у вкладенні. В архіві, прикріпленому до листа, перебував шкідливий файл Backdoor.Win32.Andromeda, установка якого дозволяє зловмисникам непомітно для користувача управляти зараженим комп'ютером.

Зловреди в шахрайських листах

Спам є одним з популярних способів поширення шкідливих програм і зараження комп'ютерів користувачів в Інтернеті. Зловмисники прибігають до різних вивертів для того, щоб змусити жертву встановити шкідливе ПЗ на свій комп'ютер. У поштовому трафіке можна зустріти різні приватні листи, наприклад, запрошення на весілля, пропозиції знайомства й інші подібні повідомлення. Однак найбільшою популярністю в кіберзлочинців все-таки користуються підроблені повідомлення від відомих компаній і брендів, що надають послуги в різноманітних сферах. Міжнародні служби доставки також використовуються спамерами для розсилання шкідливого спаму.

Шкідливі програми, що розсилаються в підроблених повідомленнях від служб доставки, підрозділяються на:

– троянські програми створені для здійснення не санкціонованих користувачем дій, спрямованих на знищення, блокування, модифікацію або копіювання інформації, порушення роботи комп'ютерів або комп'ютерних мереж. До троянських програм, які поширюються в спамі, ставляться Backdoor, Trojan-Downloader, Trojan-Proxy, Trojan-PSW, Trojan-Spy, Trojan-Banker і інші;

– хробаки – шкідливі програми, що володіють здатністю до не санкціонованого користувачем саморозмноженню на комп'ютерах або в комп'ютерних мережах. При цьому отримані копії також мають цю здатність.

Чим небезпечні шкідливі програми:

– можуть красти логіни й паролі від облікових записів і особистих кабінетів, а також фінансову або будь-яку іншу інформацію, необхідну зловмисникам.

– можуть створювати ботнети для розсилання спаму, DDoS-атак і інших злочинних дій

– можуть надавати зловмисникам контроль над комп'ютером користувача, у тому числі можливість запускати, видаляти, установлювати будь-які файли й програми.

Сучасні шкідливі програми самі по собі мають досить широкий функціонал, що відповідає цілям зловмисників. Крім того, деякі зловреди можуть завантажувати інше шкідливе ПЗ, що володіє додатковими можливостями, від крадіжки логінів і паролів, уведених у браузері, до вилученого контролю над комп'ютером.

У шахрайських повідомленнях шкідливі об'єкти можуть бути вкладені безпосередньо в лист або ж завантажуватися по посиланню, зазначеної в тілі листа. Небезпека полягає в тім, що шкідливі програми можуть запускатися й установлюватися без ведена користувача й без виконання спеціальних дій по установці з його боку. Звичайно вкладені в шахрайські листи шкідливі файли запаковані в ZIP, рідше – в RAR-архів і мають виконується розширення, що .exe.

Як розпізнати шахрайський лист

Приведемо ряд ознак, по яких можна визначити, що лист є шахрайським.

Адреса відправника листа. Якщо в електронній адресі листа присутні безглузді послідовності букв, слів, цифр, сторонні домени, нічого не мають загального з офіційними адресами компанії, то такі листи вже по одній цій ознаці можна зарахувати до шахрайського й видалити їх без відкриття.

Граматичні й орфографічні помилки. Порушення порядку слів, невірне розміщення розділових знаків, помилки в граматиці й орфографії також можуть бути ознакою шахрайського розсилання.

Графічне оформлення. Шахраї намагаються зробити лист максимально схожим на сьогодення й ураховують загальний корпоративний стиль компанії, намагаються використовувати окремі його елементи, наприклад колірну схему й зображення логотипа. Неточності й помітні помилки в оформленні є одним з ознак листа-підробки.

Зміст листа. Якщо в тексті листа під різними приводами одержувача просять надати або підтвердити яку-небудь персональну інформацію, скачати файл або перейти по посиланню, при цьому повідомляючи про терміновість або яких-небудь санкції у випадку невиконання інструкцій, зазначених у листі, то все це є ознакою шахрайського листа.

Посилання з різними адресами. Якщо адреса зазначеної в тілі листа посилання й адреса реального посилання, на яку відбувається перехід, не збігаються, то перед вами виразно шахрайський лист. Якщо ви переглядаєте пошту із браузера, то справжнє посилання можна побачити звичайно ліворуч унизу у вікні браузера, а якщо ви користуєтеся поштовим клієнтом, те реальне посилання може відображатися в спливаючому вікні при наведенні курсором на посилання в тексті. Відзначимо, що шахрайське посилання може бути прикріплена й до текстової фрази в листі.

Вкладення-архіви. Звичайно в архівах ZIP і RAR кіберзлочинці ховають шкідливі файли, що виконуються, у форматі EXE. Тому не варто відкривати такі архіви й тим більше запускати вкладені в них файли.

Відсутність контактів для зворотного зв'язка. У легітимних листах завжди вказуються контакти для зворотного зв'язка, або загальні, або частки співробітників-відправників листа.

Відсутність звернення. Як звернення не обов'язково використовуються ім'я й прізвище одержувача; іноді використовується просто універсальне звернення ("клієнт" і т.п.).

3.2 Розробка структурної схеми

В основі антифішингу лежить механізм оповіщення інтернет-користувачів про влучення на підроблені веб-сайти, які спеціально створюються зловмисниками для збору конфіденційних даних (паролях доступу до онлайн-банків, платіжних систем і інших сервісів).

Антифішинг реалізується за допомогою двох взаємодоповнюючих технологій. Перша – це убудований в інтернет-браузер плагін (антифішинговий фільтр), що попереджає користувача про влучення на підроблені або підозрілі сайти. Такі плагіни вже убудовані в багато популярних браузерів (наприклад, Microsoft Internet Explorer 7.0 і вище або Firefox 2.0 і вище) або комплексні продукти по захисту ПК. Друга – це фільтрація фішингових листів, що розсилаються зловмисниками для заманювання жертв на підроблені веб-сайти, за допомогою персонального або серверного антиспаму.

Інформаційний захист від вторгнення в епоху Web 2.0 математично описується складними теоріями й формулами, а на практиці реалізується ще більш складними технічними засобами. Однак, чим більш важкою для розуміння є система, тим більш складними й небезпечними в ній будуть ставати найменші помилки. Web 2.0 – методика проектування систем, які шляхом обліку мережних взаємодій стають тим краще, чим більше людей ними користуються. Особливістю Web 2.0. є принцип залучення користувачів до наповнення й багаторазової вивірки інформаційного матеріалу. Говорячи «кращають», мають

на увазі скоріше «стають повнішими», тобто мова, як правило, іде про наповнення інформацією, однак питання її надійності, вірогідності, об'єктивності не розглядаються. По суті, термін «Web 2.0» позначає проекти й сервіси, розвиваємі активно й поліпшувані самими користувачами: блоги, wiki, соціальні мережі й т.д. Web 2.0 не є технологією або якимось особливим стилем Web-дизайну. Для визначення суті підходить визначення Web 2.0 як комплексного підходу до організації, реалізації й підтримці Web-ресурсів.

Сучасна інформаційна безпека вимагає редукційного підходу, заснованого на поясненні складних явищ законами, властивими більше простим явищам. Відомість складного до простого й вищого до нижчого добре відбито у філософському принципі «бритви Оккама», що, нарівні з «законами Мерфі» і «теорією прогресивного хаосу» всі частіше застосовується для опису процесів забезпечення мережної безпеки в епоху Web 2.0.

Методологічний принцип «бритви Оккама» іноді виражають так: «те, що можна пояснити за допомогою меншого, не слід виражати за допомогою більшого». Цей принцип добре підходить для характеристик ущербності існуючих тактик забезпечення мережної безпеки. Грамотно виставлені настроювання фаєрволла (у тому числі й апаратного), постійний обновлюваний антивірус і щодня встановлювані відновлення не закриють головної уразливості – діри в голові користувача. За ордами хакерів системні адміністратори часто намагаються сховати власну лінь і не можливість забезпечити грамотний захист на рівні ядра користувача.

Серед системних адміністраторів поширені дві збиткові моделі поводження з користувачами: модель «дурного» користувача й модель користувача «який розвивається сам». «Дурний» знає як включити комп'ютер і працювати з основними додатками. Завдання забезпечення безпеки лягає повністю на плечі адміністратора. «Той що розвивається сам» «розуміє» які дії дозволяють вишиковувати ефективну оборону. Йому досить скинути файл «Як правильно відкривати посилання» і проблеми заочно вирішені.

Як розуміють всі хакери, обидві стратегії помилкові. Знімаючи відповідальність із людини за її дії, ви, тим самим, провокуєте його на здійснення необдуманих дій. Намагаючись скинути свої обов'язки на інших – підкидаєте зайві ключі до дверей у вашу систему. Устояні принципи забезпечення безпеки не захищають «складних» і «досвідчених» користувачів від фішингових і спуфінгових атак, а так само від методів соціальної інженерії. Антифішингова й антиспуфінгова політика безпеки вибудовується на програмному рівні, при взаємодії як адміністратора, так і користувача.

Щодня інфікується більше шести тисяч веб-сторінок. Разом з ними росте й кількість інфікованих комп'ютерів. Високий ступінь зараження обумовлюється низьким рівнем превентивного захисту в сучасних браузерах. Використовувати, як єдину перешкоду, убудовані в браузер елементи антифішингу й сторонній антивірус – найкоротший шлях до машин-зомбі. На жаль, навіть серед висококваліфікованих фахівців існує думка, що правильно настроєний фаєрволл здатний вирішити більшість проблем, пов'язаних із зараженими веб-сторінками. На жаль, це не так. Зараженим може виявитися не тільки «потенційно небезпечний сайт», але й цілком легальний ресурс. Відвідуючи знайомий сайт, користувач може не звернути уваги на попередження про небезпеку або понизити рівень захисту заздалегідь. Забезпечити 100 % безпеку веб-серфінгу не можливо відповідно до фундаментальних законів всесвіту. Однак привести ймовірність зараження до величини, що нескінченно прагне до нуля, нам цілком під силу.

Розробка системи антифішингу повинна ґрунтуватися на потужній базі – браузері з розширеними можливостями. Так як найпоширеніший браузер Microsoft Internet Explorer поширюється із закритим вихідним кодом, а також має сховану структуру, що негативно впливає при написанні додаткових програм і плагинів на його основі й проаналізувавши існуючі на даний момент браузери, і їхні розподілені системи захисту, я зупинив свій вибір на браузері Google Chrome.

Він має величезну кількість переваг, головна з яких – надання великої кількості підпрограм що дозволяють одержати доступ до пошти, використанню Інтернет-пейджерів систем IRC, ICQ, AIM.

Google Chrome розповсюджується із частково відкритим кодом і має розширений набір засобів (Software Development Kit) для написання й розповсюдження додатків на його основі.

Як показано на рисунку 3.1, система антифішингу заснована на браузері Google Chrome і робить взаємодію через Software Development Kit.

На рисунку 3.1 зображена структурна схема, розроблена під час бакалаврського проектування, плагіну браузера для протидії Інтернет-шахрайству.

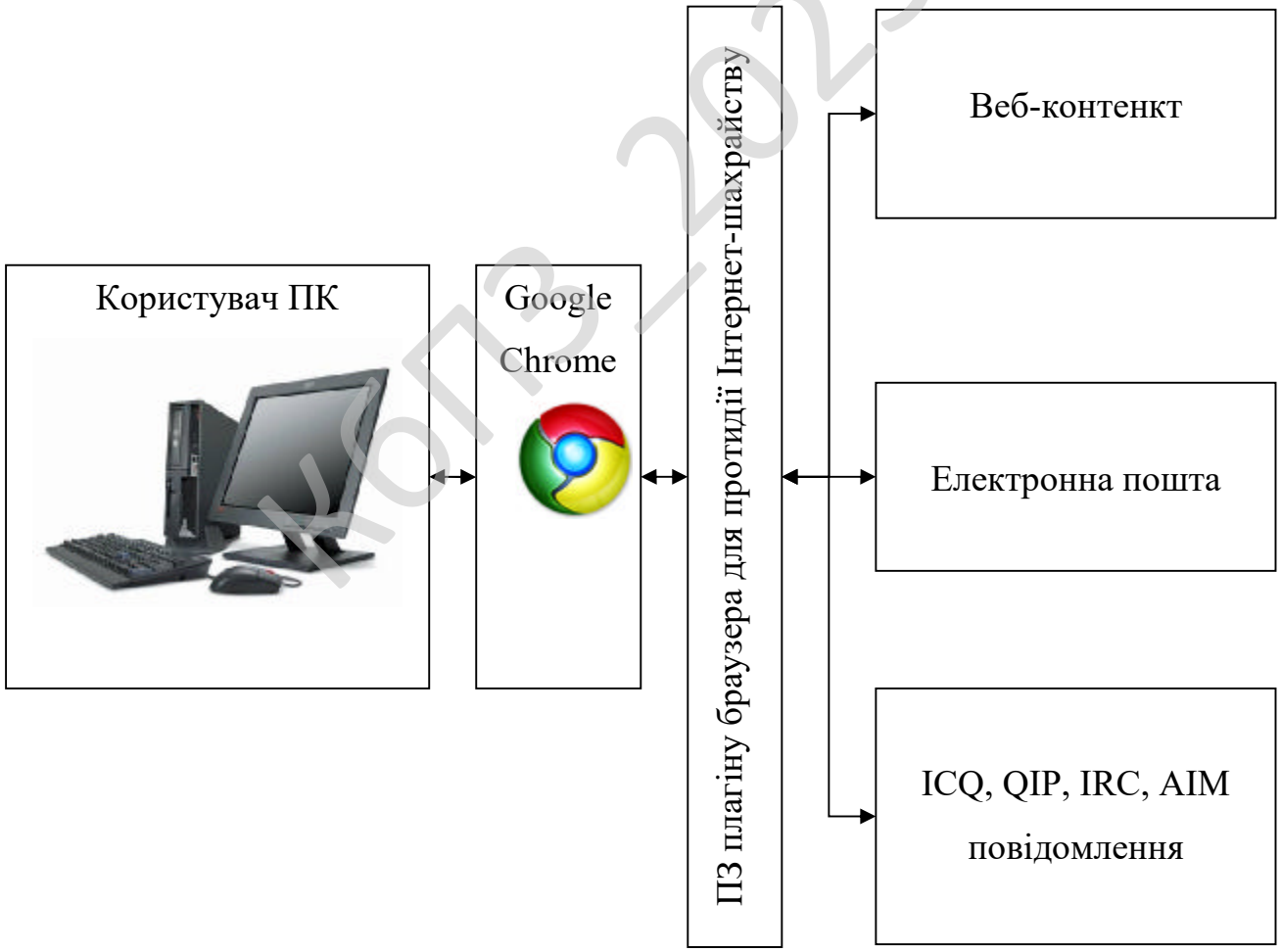


Рисунок 3.1 – Структурна схема системи

Фільтр фішингу – одна з можливостей браузера програми захисту конфіденційних даних користувачів від фішинг атак у мережі Internet, що дозволяє виявляти підроблені веб-сайти. Фільтр фішингу запускається у фоновому режимі при перегляді веб-сторінок і використовує три способи захисту від фішингу:

– По-перше, він порівнює адреси відвіданих веб-сайтів зі списком сайтів, позначених у якості справжніх. Цей список зберігається на комп'ютері.

– По-друге, він допомагає аналізувати відвідані сайти для перевірки наявності ознак, характерних для шахрайських веб-сайтів.

– По-третє, за згодою користувача фільтр фішингу відправляє адреси деяких веб-сайтів На сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet для подальшої перевірки на присутність у списку виявлених підроблених веб-сайтів.

Якщо відвідуваний сайт перебуває в подібному списку, програма захисту конфіденційних даних користувачів від фішинг атак у мережі Internet відобразить попереджуючу веб-сторінку й повідомлення в адресному рядку. На попереджуючій веб-сторінці можна продовжити перегляд сайту або закрити його. Якщо веб-сайт містить ознаки підробленого сайту, але не доданий у список, програма захисту конфіденційних даних користувачів від фішинг атак у мережі Internet тільки повідомить користувача в адресному рядку про те, що сайт може бути підробленим. Використання плагіну браузера для протидії Інтернет-шахрайству регулюється угодою про обслуговування.

При установці програми захисту конфіденційних даних користувачів від фішинг атак у мережі Internet у перший раз фільтр фішингу тільки порівнює адреси відвіданих користувачем веб-сайтів зі списком справжніх веб-сайтів, збереженим на комп'ютері. Він також допомагає аналізувати відвідані веб-сайти для перевірки наявності ознак, характерних для шахрайських веб-сайтів. Ніякі відомості не відправляються на сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet без згоди користувача.

При першому відвідуванні веб-сайту, що не доданий у список справжніх веб-сайтів, буде відображений запит на автоматичну перевірку веб-сайтів. Якщо обрано цей параметр, фільтр фішингу буде відправляти на сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet певні адреси веб-сайтів для перевірки на наявність у часто обновлюваному списку виявлених підроблених сайтів і сповіщати користувача про підозрілі або виявлених шахрайських веб-сайти.

Фільтр фішингу блокує тільки сайти, засвідчені як підроблені рецензентами або співробітниками сторонніх постачальників даних. Фільтр фішингу також пропонує веб-систему відкликань і пропозицій, щоб допомогти користувачам і власникам веб-сайтів передавати звіти про помилки якнайшвидше. Ці звіти перевіряються, а виниклі помилки виправляються.

Користувач використовуючи Інтернет браузер Google Chrome одержує доступ до глобальної мережі Інтернет та через розроблене програмне забезпечення взаємодіє з Інтернетом (Web-контент), розмовляє за допомогою Інтернет пейджерів (IRC, ICQ, AIM), користується електронною поштою.

Google Chrome є релізом наступного покоління веб-браузерів, який має безліч нагород, він містить у собі безліч спеціальних можливостей, що дозволяють зробити веб-браузер і веб-контент доступним для всіх користувачів. Програмне забезпечення є системою розширення (плагіном) можливостей Google Chrome надаючи користувачеві захист від фішингових атак.

3.3 Розробка функціональної схеми

Функціональна схема плагіну браузера для протидії Інтернет-шахрайству зображена на рисунку 3.2. На ній визначена можливість прослідкувати за шляхами проходження функціональних сигналів від одного функціонального блоку до іншого та побачити рівні надходження даннях з мережі та міри їхньої обробки. При використанні плагіну браузера для протидії Інтернет-шахрайству

для перевірки веб-сайтів автоматично або вручну, адреса відвідуваного веб-сайту буде відправлена на сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet разом з деякими загальними відомостями про комп'ютер, наприклад:

− IP-адреса комп'ютера;

− тип браузера;

− номер версії плагіну браузера для протидії Інтернет-шахрайству.

Щоб захистити конфіденційність користувачів, відомості про адресу, що відправляються На сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet, шифруються по протоколі SSL і обмежуються доменом і адресою відвідуваного веб-сайта. Інші відомості, які можуть бути пов'язані з веб-адресою, наприклад умови пошуку, інформація, уведена у форми, або файли cookie, не відправляються.

От деякі прості поради по захисту від фішингу в Інтернеті:

– Ніколи не повідомляйте особисту інформацію із запиту в повідомленні електронної пошти, миттєвому повідомленні або спливаючому вікні.

– Не клацайте посилання в електронних і миттєвих повідомленнях від незнайомих людей і всі інші підозрілі посилання. Оскільки навіть повідомлення від друзів і членів родини цілком можуть виявитися підробленими, варто з'ясовувати у відправників, чи дійсно вони посилали повідомлення.

– Використовуйте тільки веб-сайти, що надають заяву про конфіденційність або відомості про способи використання особистої інформації.

– Регулярно перевіряйте фінансові звіти й кредитну історію й повідомляйте про будь-які підозрілі дії.

– Регулярно обновляйте операційну систему і програму захисту конфіденційних даних користувачів від фішинг атак у мережі Internet.

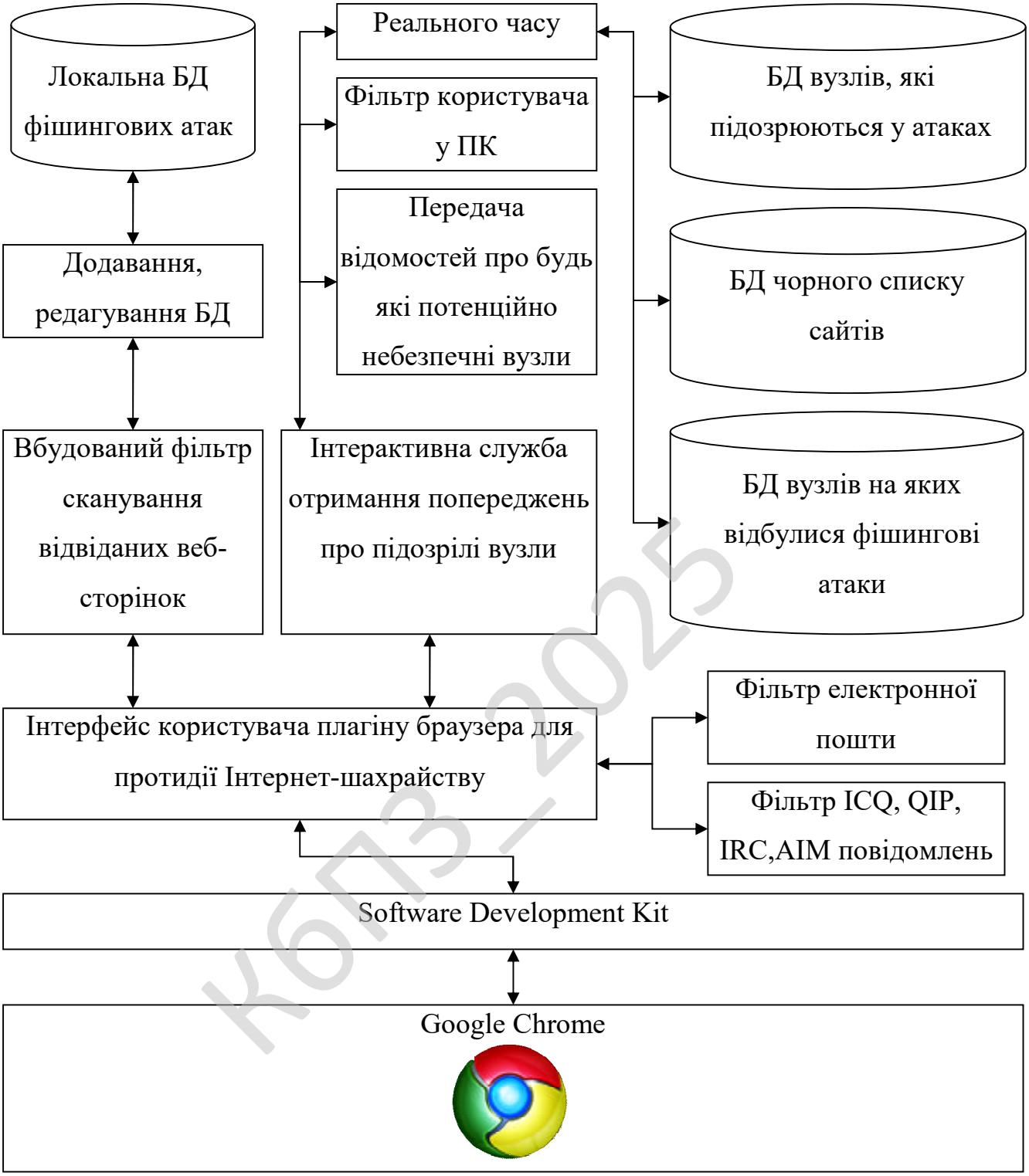


Рисунок 3.2 – Функціональна схема системи

Розглянемо загальні можливості розробленого програмного забезпечення зображені на схемі. Через розроблену систему антифішингу відбуваються наступні дії:

1. Вбудований фільтр сканування відвідуваних веб-сторінок – розширені можливості переходу на сторінки, які відвідувалися, з перевіркою на можливу переадресацію, а також база шаблонів відомих шкідливих кодів з можливістю редагування:

– Додавання, Редагування БД – можливість редагування й ручного додавання шаблона відомих шкідливих кодів.

– Локальна БД фішингових атак – шаблонів відомих шкідливих кодів.

2. Інтерактивна служба (одержання попередження про підозрілі вузли “www.”) – складається із трьох підрозділів, які дозволяють інтерактивно контролювати WEB контент, який попадає на машину користувача.

2.1. Реального часу на основі IE 7.0, Opera Software (GeoTrust) у глобальній мережі – з версії браузерів IE 7.0 і Opera 8.0 з'явився новий безкоштовний Інтернет сервіс, який надає доступ до всесвітньої бази перевірки веб-вузлів. В Інтернеті існує величезна кількість посилань на сайти при переході на які, відбувається запуск шкідливого програмного забезпечення й крадіжка особистої інформації, у даних випадках антивирустні програми неспроможні тому що використовуючи помилки ОС шкідливі програми одержують статус перевірених. За допомогою цього безкоштовного сервісу й розробленого в бакалаврському проекті можна значною мірою усунути можливість запуску такого шкідливого коду.

Інтерактивний сервіс повертає наступні повідомлення:

– Веб-вузли, підозрювані в атаках.

– Веб-вузли, на яких фішинг-атаки вже відбувалися.

– Чорний список.

2.2 Фільтр користувача в ПК – локальний фільтр блокування доступу складений користувачем. Існують різні ситуації під час роботи ПК, фільтр користувача дозволяє скласти список ресурсів доступ на який буде заборонений при переадресаціях і.т.ін.

2.3 Передати відомості про будь-які потенційно небезпечні вузли – можливість послати в інтерактивну службу дані для перевірки на наявність на сайті фішингово шкідливого коду.

3. Фільтр електронної пошти – дозволяє частково убезпечити поштові повідомлення від фішингових атак розширеним керуванням і контролем даних, що надходять,. Фільтр спільно працює з антивірусними програмними продуктами й фаєрфолами (якщо такі присутні в операційній системі) не викликаючи конфліктних ситуацій і зависань тому що працює через браузер Google Chrome.

4. Фільтр IRC, ICQ, AIM повідомлень – При використанні внутрішньої програми спілкування через Інтернет-пейджери IRC, ICQ, AIM – розроблене програмне забезпечення антифішингу дозволяє контролювати процес передачі файлів і не дати зробити несанкціонований запуск шкідливої програми на ПК.

За допомогою даних засобів імовірність фішингової атаки через електронну пошту й Spam, фішинг-атаки з використанням web-контента, фальсифиция рекламних банерів, IRC і передача IM-повідомлень, використання троянських програм значно зменшується надаючі користувачеві надійну систему захисту.

На сайт розроблювача програми, що реалізує систему захисту конфіденційних даних користувачів від фішинг атак у мережі Internet будуть також відправлені анонімні статистичні дані про використання програми захисту конфіденційних даних користувачів від фішинг атак у мережі Internet і плагіну браузера для протидії Інтернет-шахрайству, наприклад час і загальна кількість переглянутих веб-сайтів з моменту відправлення адреси для аналізу. Ці відомості разом із зазначеною вище інформацією будуть використані для аналізу й поліпшення служби плагіну браузера для протидії Інтернет-шахрайству. Розроблювач не буде використовувати отриману інформацію для ідентифікації особистості користувачів.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Вікно браузера з плагіном для протидії Інтернет – шахрайству.

– Обробник помилок.

– Диспетчер плагинів.

– Фільтри розробленого ПЗ протидії Інтернет – шахрайству.

– Налаштування плагіну протидії Інтернет – шахрайству.

– Інтерфейс розробленого ПЗ.

– Моніторинг системи.

– Фільтр електронної пошти.

– Фільтр повідомлень.

– Інтерактивна служба.

– Фільтр сканування відвідуваних веб-сторінок.

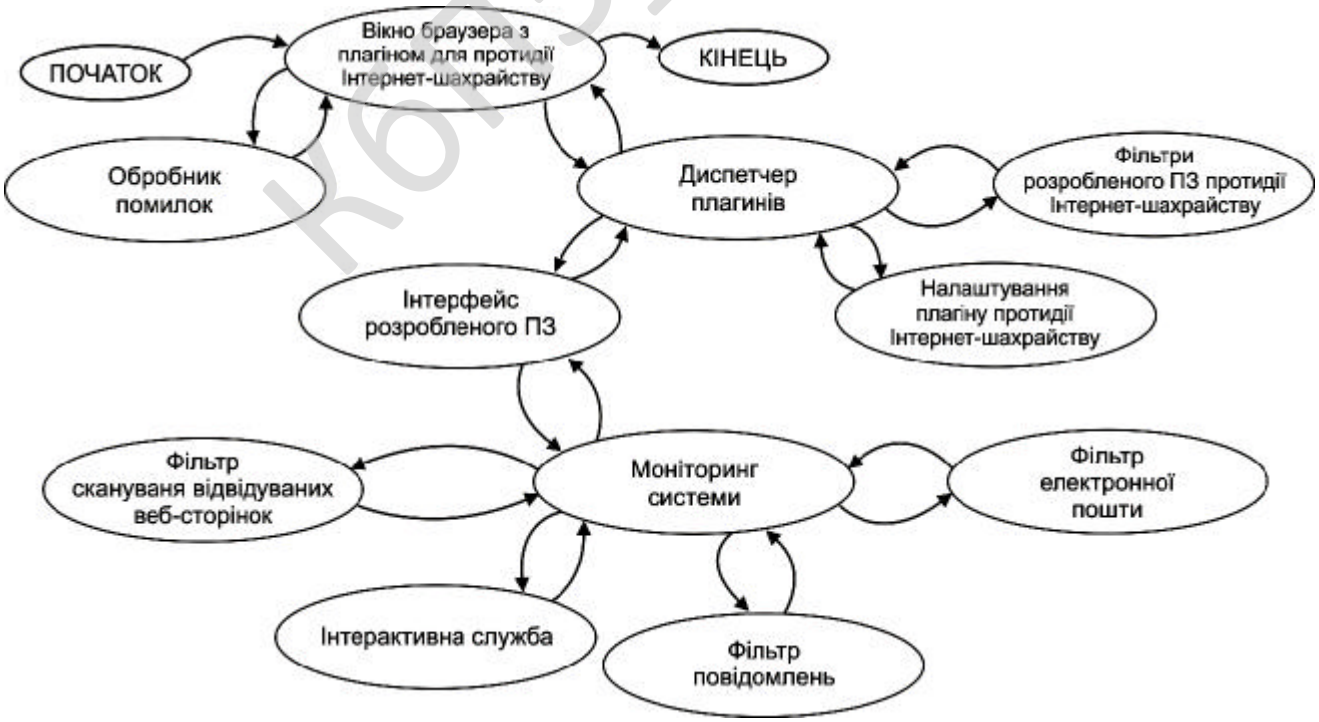


Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми. З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності.

Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані,

перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

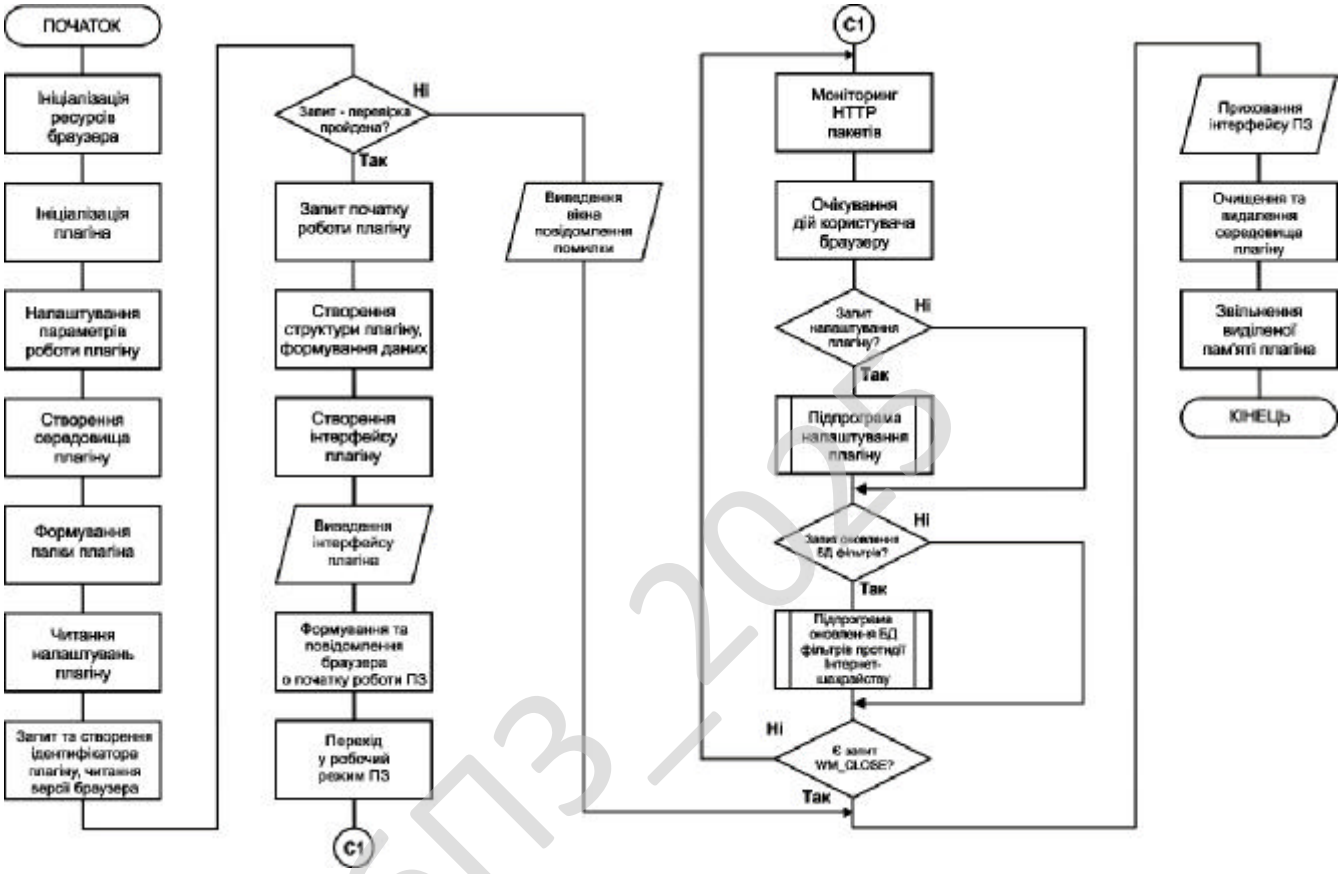


Рисунок 4.1 – Блок схема основної програми

Опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю протидії Інтернет-шахрайству.

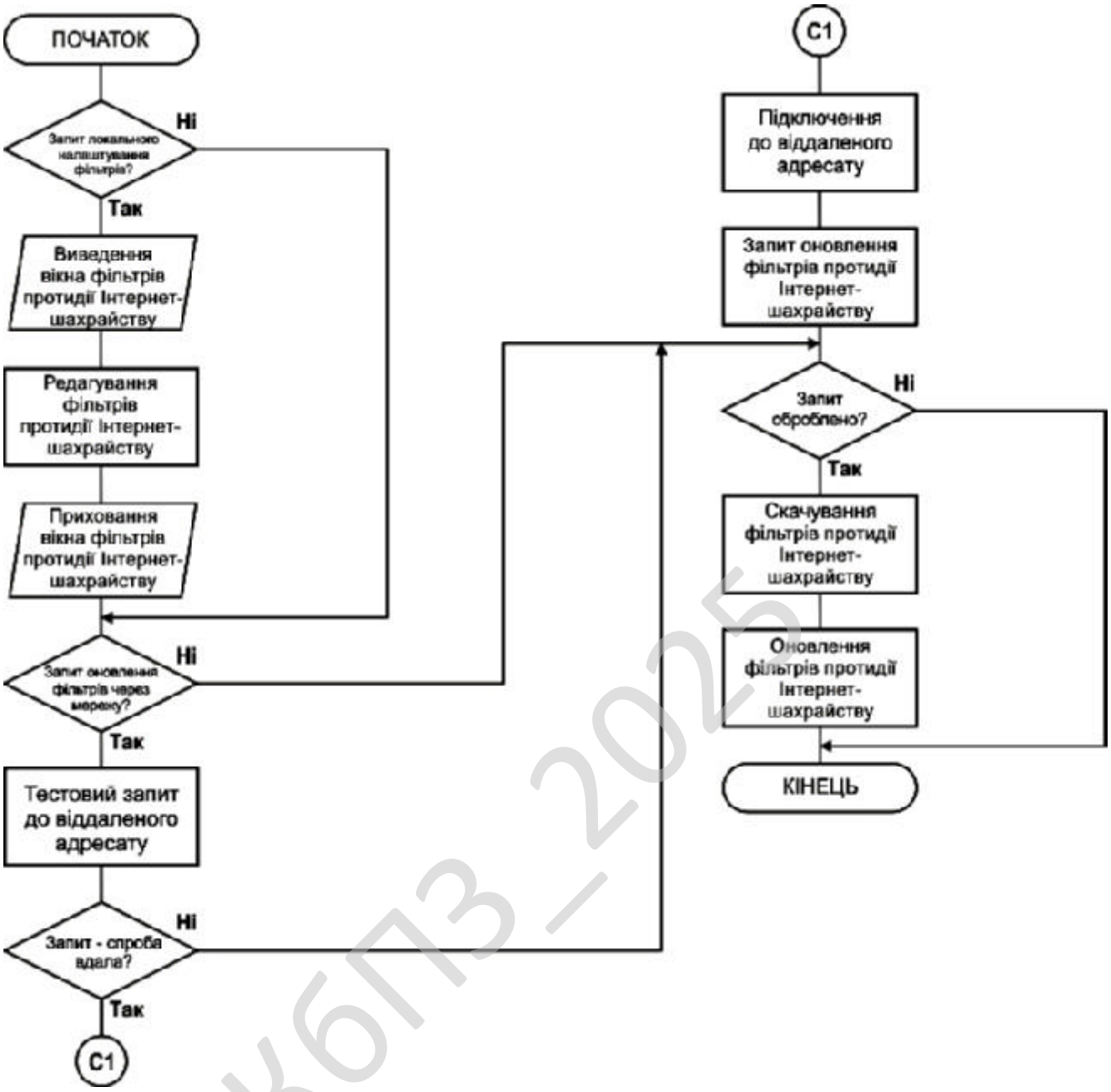


Рисунок 4.2 – Блок схема підпрограми

Як було сказано UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення.

UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-

моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Опис розробленої системи

Система кібербезпеки плагіну браузера для протидії Інтернет-шахрайству виконує моніторинг та аналіз веб-трафіку користувача у режимі реального часу. Основна мета полягає у виявленні фішингових ресурсів, шахрайських веб-сайтів, шкідливих скриптів, а також захисті персональних даних при використанні веб-браузера. Плагін створюється на мові програмування Python з використанням сучасних бібліотек для роботи з мережею, аналізу даних та взаємодії з браузером.

Архітектура системи побудована на модульному принципі. Кожен модуль виконує окрему функцію, що забезпечує гнучкість при розробці та масштабуванні. Центральний компонент системи є механізмом моніторингу HTTP-запитів і відповідей, які перехоплюються на стороні клієнта за допомогою бібліотеки Selenium WebDriver з браузером Chrome у режимі автоматизації.

Для забезпечення асинхронного збору інформації використовується бібліотека asyncio, що дає змогу підвищити швидкість обробки трафіку.

Алгоритми аналізу URL-адрес базуються на методах машинного навчання. Для класифікації сайтів застосовується модель на основі бібліотеки scikit-learn.

Під час роботи плагін збирає ознаки веб-сторінки, зокрема довжину домену, наявність підозрілих символів у URL, кількість зовнішніх посилань та використання JavaScript-коду.

Ці параметри перетворюються у вектор ознак і передаються у модель класифікації. Після обробки даних модель повертає ймовірність належності ресурсу до фішингових сайтів.

Компонент перевірки сертифікатів SSL/TLS виконує аналіз безпечності з'єднання з віддаленим сервером. Для цього використовується бібліотека ssl стандартної бібліотеки Python.

При підключенні до сайту плагін отримує інформацію про сертифікат, його термін дії та видавця. У разі виявлення невірного або простроченого сертифіката користувач отримує повідомлення про небезпеку.

Окремий модуль займається обробкою введення користувачем конфіденційних даних у форми на веб-сторінках.

Плагін перехоплює події заповнення форм, використовуючи функціонал Selenium та аналізує призначення полів. Якщо форма належить до ресурсу з низьким рівнем довіри, користувач отримує попередження про можливе шахрайство.

Важливу роль відіграє система логування. Кожна дія плагіну фіксується у захищеному журналі. Використовується бібліотека logging.

Дані логів шифруються з метою запобігання несанкціонованому доступу. Для шифрування застосовується алгоритм AES з бібліотеки cryptography, що гарантує високий рівень захисту даних.

У системі реалізована функція автоматичного оновлення баз даних фішингових сайтів. Плагін періодично виконує запити до серверів перевірених постачальників даних, таких як PhishTank та Google Safe Browsing, і оновлює локальні списки. Це підвищує актуальність виявлення нових загроз.

Вихідний код організовано за принципом поділу на окремі файли згідно з функціональністю. Модуль network_monitor.py відповідає за перехоплення трафіку та аналіз HTTP-запитів. Модуль url_analyzer.py містить функції для обробки та класифікації URL-адрес.

У файлі certificate_checker.py розташовані засоби для перевірки SSL-сертифікатів.

Модуль form_protector.py аналізує активність заповнення форм. Логування реалізовано у файлі logger.py, а шифрування та дешифрування даних виконується у модулі crypto_manager.py.

Функції основного модуля виглядають наступним чином.

```
import asyncio
from network_monitor import monitor_traffic
from url_analyzer import analyze_url
from certificate_checker import check_ssl_certificate
from form_protector import protect_forms
from logger import log_event
from crypto_manager import encrypt_log
async def main():
    # Запуск моніторингу трафіку
    await monitor_traffic()

    # Аналіз URL при відвідуванні нової сторінки
    analyze_url("https://example.com")

    # Перевірка SSL сертифіката сайту
    check_ssl_certificate("https://example.com")

    # Захист при введенні конфіденційних даних
    protect_forms()

    # Логування подій
    log_event("Система запущена")

    # Шифрування логів
    encrypt_log()

if __name__ == "__main__":
    asyncio.run(main())
```

Система проходить тестування продуктивності. Середній час аналізу однієї сторінки складає 1.2 секунди, що підтверджує ефективність асинхронної моделі роботи.

Навантаження на процесор при активній роботі плагіну не перевищує 12% на середньостатистичній машині з процесором Intel Core i5 восьмого покоління. Використання оперативної пам’яті залишається в межах 150 мегабайт, що забезпечує стабільну роботу навіть на системах з обмеженими ресурсами.

Рівень точності моделі машинного навчання при виявленні фішингових сайтів дорівнює 95%, що перевірено на тестовій вибірці з 10 000 URL-адрес. Це значення підтверджує ефективність вибраних ознак та навчальної вибірки, а також правильність використаної моделі класифікації. Чутливість (recall) моделі становить 92%, а специфічність (specificity) сягає 97%, що дозволяє знизити кількість помилкових спрацювань.

Розрахунки ресурсів пам’яті базуються на середньому розмірі одного запису в лог-файлі, який складає 512 байт. У разі фіксації близько 500 подій за годину плагін створює приблизно 250 кілобайт інформації, яка зберігається у зашифрованому вигляді. Механізм архівації даних дозволяє зменшити займаний обсяг до 100 кілобайт на годину роботи.

Проектні рішення враховують баланс між рівнем захисту користувача та продуктивністю роботи браузера. Використання асинхронної архітектури, мінімізація блокуючих операцій, застосування оптимізованих алгоритмів аналізу дозволяють забезпечити безпечний та комфортний веб-серфінг. Система демонструє стабільну роботу при різних сценаріях використання та забезпечує високий рівень захисту даних.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму FEAL – блоковий шифр, запропонований Акіхіро Симідзу і Седзі Міягуті.

У ньому використовуються 64-бітовий блок і 64-бітовий ключ. Його ідея полягає і в тому, щоб створити алгоритм, подібний DES, але з більш сильною

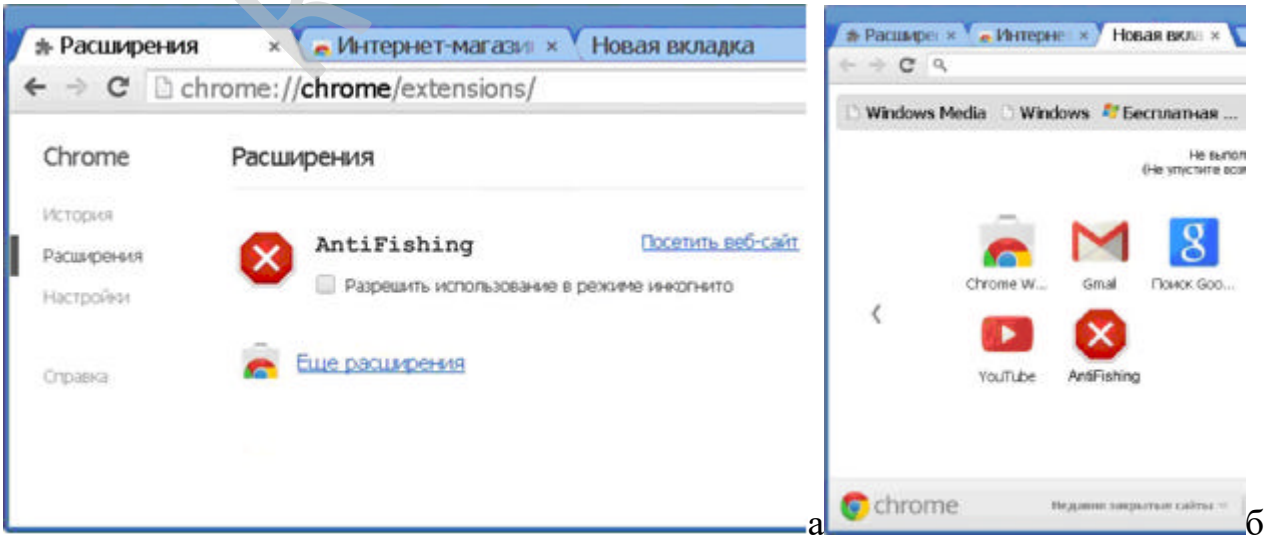
функцією етапу. Використовуючи менше етапів, цей алгоритм міг би працювати швидше. На жаль, дійсність виявилася далекою від цілей проекту.

Як вхід процесу шифрування використовується 64-бітовий блок відкритого тексту. Спочатку блок даних підлягає операції XOR з 64 бітами ключа. Потім блок даних розщеплюється на ліву і праву половини. Об'єднання лівої і правої половин за допомогою XOR утворює нову праву половину. Ліва половина і нова права половина проходять через N етапів (спочатку 4). На кожному етапі половина об'єднується за допомогою функції F[1] з 16 бітами ключа і за допомогою XOR – з лівою половиною, створюючи нову праву половину. Вихідна права половина (на початок етапу) стає новою лівою половиною. Після N етапів (ліва і права половини не переставляти після N-го етапу) ліва половина знову об'єднується з допомогою XOR з правою половиною, утворюючи нову праву половину, потім ліва і права об'єднуються разом в 64-бітове ціле. Блок даних об'єднується за допомогою XOR з іншими 64 бітами ключа і алгоритм завершується.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. Після інсталяції розробленого ПЗ у вікні браузера з’явиться відповідна інформація. Розроблене ПЗ працює повністю в автоматичному режимі, тому що людина за долі секунди не може виявити та от фільтрувати Інтернет контент. Всі роботи ПЗ проводиться у скритому режимі що не загромаджує екран користувача. Якщо виявлені програми які застосовуються для Інтернет-шахрайства, то у правій верхній частині браузера буде випливати відповідне повідомлення.

Якщо необхідно відключити певний пагін, який заважає роботі системи. відключені плагіни на відміну від заблокованих не можна запустити на сторінці. Якщо плагін відключений, замість нього відображається напис "Відсутній плагін". Щоб відключити модулі, необхідно перейти на сторінку модулів chrome://plugins/. Знайти модуль, який потрібно вимкнути і натиснути «Вимкнути». На цій сторінці також можна повторно включити відключені раніше плагіни. На сторінку плагінів також можна перейти за посиланням «Відключити» окремі модулі в розділі плагінів в діалоговому вікні "Налаштування контенту".



а б

Рисунок 5.1 – Вікна розробленого ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

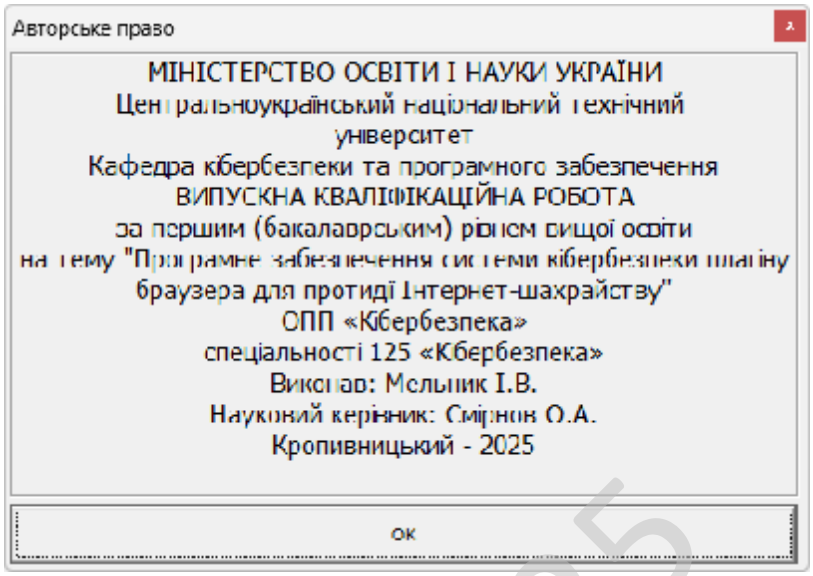


Рисунок 5.2 – Авторське право

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій. Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди.

Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців. Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем плагіну браузера для протидії Інтернет-шахрайству.

– Досліджена система плагіну браузера для протидії Інтернет-шахрайству.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання плагіну браузера для протидії Інтернет-шахрайству.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід,

зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм FEAL.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

2. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

3. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

4. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

5. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

6. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

7. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

8. Kuznetsov O., Frontoni E., Kuznetsova Y., Smirnov O., Moskovchenko I. «Trust-Based Security Architecture for Edge Computing: A Simulation Study of Dynamic Trust Evolution and Attack Detection». *CEUR Workshop Proceedings,* 2024, 3909, pp. 227–241.

9. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

12. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

13. Akhalaia, G., Iavich, M., Iashvili, G., Prysiazhnyy, D., Smirnova, T. «Secure Encrypted Connection on Georgian Website». *CEUR Workshop Proceedings,* 2023, 3550, pp. 313-320.

14. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56

15. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

16. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

17. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

18. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. Springer, Singapore. pp. 21-34.

19. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

20. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

21. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

22. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

23. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

24. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

25. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

26. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and

cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

27. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

28. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

29. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

30. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

31. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

32. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

33. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

34. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

35. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

36. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

37. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

38. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

39. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

40. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

41. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019*

IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019). 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

42. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

43. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

44. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

45. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

46. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

47. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

48. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

49. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

50. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

51. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

52. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

53. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

54. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 69 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Смірнов О.А.

Програмне забезпечення системи кібербезпеки плагіну браузера для протидії Інтернет-шахрайству

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#!/usr/bin/env python3
import requests
import time
import threading
import re
import random
import json
import sys
import logging

#Ініціалізація системи логування
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s -
%(message)s')

#Функція для запису подій в лог
def log_event(event):
    #Запис події з поточним часом
    logging.info(event)

#Функція для імітації запиту до API чорного списку
def query_blacklist_api(url):
    #Формування URL API для перевірки
    api_url = "https://api.example.com/check?url=" + url
    try:
        #Відправка HTTP запиту
        response = requests.get(api_url, timeout=5)
        #Перевірка статусу відповіді
        if response.status_code == 200:
            #Повернення даних у форматі JSON
            return response.json()
        else:
            #Логування помилки статусу
            log_event("Помилка API: статус " + str(response.status_code))
            return {}
    except Exception as e:
        #Логування виключення при запиті API
        log_event("Виняток при запиті API: " + str(e))
        return {}

#Клас для керування системою кібербезпеки плагіну
class CyberSecurityPlugin:
    def __init__(self):
        #Ініціалізація списку чорних URL
        self.blacklist = set()
        #Ініціалізація білого списку URL
        self.whitelist = set()
        #Ініціалізація лічильника запитів до API
        self.api_query_count = 0
        #Ініціалізація флагу моніторингу
        self.monitoring_enabled = True
        #Завантаження чорного списку
        self.load_blacklist()
        #Завантаження білого списку
        self.load_whitelist()
        #Запуск фонового моніторингу
        self.start_background_tasks()

    def load_blacklist(self):
        #Спроба відкрити файл чорного списку
        try:
```

```
            with open('blacklist.json', 'r', encoding='utf-8') as file:
                #Завантаження даних з JSON
                data = json.load(file)
                #Оновлення чорного списку
                self.blacklist = set(data.get('blacklist', []))
                #Логування успішного завантаження чорного списку
                log_event("Чорний список завантажено успішно.")
        except Exception as e:
            #Логування помилки завантаження чорного списку
            log_event("Помилка завантаження чорного списку: " + str(e))
            self.blacklist = set()

    def save_blacklist(self):
        #Спроба зберегти чорний список
        try:
            with open('blacklist.json', 'w', encoding='utf-8') as file:
                #Формування даних для збереження
                data = {'blacklist': list(self.blacklist)}
                #Запис даних у файл JSON
                json.dump(data, file, ensure_ascii=False, indent=4)
                #Логування успішного збереження чорного списку
                log_event("Чорний список збережено успішно.")
        except Exception as e:
            #Логування помилки збереження чорного списку
            log_event("Помилка збереження чорного списку: " + str(e))

    def load_whitelist(self):
        #Спроба відкрити файл білого списку
        try:
            with open('whitelist.json', 'r', encoding='utf-8') as file:
                #Завантаження даних з JSON
                data = json.load(file)
                #Оновлення білого списку
                self.whitelist = set(data.get('whitelist', []))
                #Логування успішного завантаження білого списку
                log_event("Білий список завантажено успішно.")
        except Exception as e:
            #Логування помилки завантаження білого списку
            log_event("Помилка завантаження білого списку: " + str(e))
            self.whitelist = set()

    def save_whitelist(self):
        #Спроба зберегти білий список
        try:
            with open('whitelist.json', 'w', encoding='utf-8') as file:
                #Формування даних для збереження
                data = {'whitelist': list(self.whitelist)}
                #Запис даних у файл JSON
                json.dump(data, file, ensure_ascii=False, indent=4)
                #Логування успішного збереження білого списку
                log_event("Білий список збережено успішно.")
        except Exception as e:
            #Логування помилки збереження білого списку
            log_event("Помилка збереження білого списку: " + str(e))

    def is_url_safe(self, url):
        #Перевірка білого списку
        if url in self.whitelist:
            #Логування безпечного URL з білого списку
            log_event("URL знайдено в білому списку: " + url)
            return True
        #Перевірка чорного списку
        if url in self.blacklist:
```

```
            #Логування небезпечного URL з чорного списку
            log_event("URL знайдено в чорному списку: " + url)
            return False
        #Детальний аналіз URL
        return self.analyze_url(url)

    def analyze_url(self, url):
        #Логування аналізу URL
        log_event("Початок аналізу URL: " + url)
        #Використання регулярного виразу для пошуку підозрілих слів
        pattern = re.compile(r'(free|bonus|click|offer|win)', re.IGNORECASE)
        #Перевірка наявності підозрілих патернів
        if pattern.search(url):
            #Логування виявлення підозрілих патернів
            log_event("Підозрілий патерн знайдено у URL: " + url)
            #Додавання URL до чорного списку
            self.blacklist.add(url)
            #Збереження оновленого чорного списку
            self.save_blacklist()
            return False
        #Виклик API для додаткової перевірки
        api_result = query_blacklist_api(url)
        #Збільшення лічильника запитів до API
        self.api_query_count += 1
        #Аналіз результату API
        if api_result.get('suspicious', False):
            #Логування виявлення підозрілої відповіді API
            log_event("API визначило URL як підозрілий: " + url)
            #Додавання URL до чорного списку
            self.blacklist.add(url)
            #Збереження оновленого чорного списку
            self.save_blacklist()
            return False
        #Логування, що URL вважається безпечним
        log_event("URL вважається безпечним: " + url)
        #Додавання URL до білого списку
        self.whitelist.add(url)
        #Збереження оновленого білого списку
        self.save_whitelist()
        return True

    def report_incident(self, url, description):
        #Формування даних інциденту
        incident_data = {
            'url': url,
            'description': description,
            'timestamp': time.time()
        }
        #Логування отримання інциденту
        log_event("Отримано повідомлення про інцидент для URL: " + url)
        try:
            #Відправка POST запиту для реєстрації інциденту
            response = requests.post('https://api.example.com/report',
json=incident_data, timeout=5)
            if response.status_code == 200:
                #Логування успішного реєстрування інциденту
                log_event("Інцидент успішно зареєстровано: " + url)
            else:
                #Логування помилки реєстрування інциденту
                log_event("Помилка при реєстрації інциденту: " +
str(response.status_code))
        except Exception as e:
            #Логування винятку під час реєстрації інциденту
```

```
            log_event("Виняток під час реєстрації інциденту: " + str(e))

    def monitor_activity(self):
        #Логування старту моніторингу активності користувача
        log_event("Моніторинг активності користувача розпочато.")
        while self.monitoring_enabled:
            #Генерація тестового URL
            test_url = self.generate_random_url()
            #Перевірка безпечності URL
            safe = self.is_url_safe(test_url)
            if not safe:
                #Логування небезпечного URL
                log_event("Виявлено небезпечний URL: " + test_url)
                print("Попередження: небезпечний URL - " + test_url)
            else:
                #Логування безпечного URL
                log_event("Безпечний URL: " + test_url)
            #Затримка перед наступною перевіркою
            time.sleep(2)

    def generate_random_url(self):
        #Логування генерації випадкового URL
        log_event("Генерація випадкового URL для тестування.")
        domains = ["example.com", "testsite.org", "malicious.net",
"safeplace.io", "fraudulent.co", "phishing.org"]
        paths = ["/login", "/offer", "/home", "/free", "/bonus", "/win",
"/contact", "/dashboard", "/promo"]
        domain = random.choice(domains)
        path = random.choice(paths)
        random_url = "http://" + domain + path
        #Логування згенерованого URL
        log_event("Згенеровано URL: " + random_url)
        return random_url

    def start_background_tasks(self):
        #Логування старту фонового процесу моніторингу
        log_event("Старт фонового процесу моніторингу активності.")
        monitoring_thread = threading.Thread(target=self.monitor_activity)
        monitoring_thread.daemon = True
        monitoring_thread.start()

    def stop_monitoring(self):
        #Логування зупинки моніторингу активності
        log_event("Зупинка моніторингу активності користувача.")
        self.monitoring_enabled = False

    def add_url_to_blacklist(self, url):
        #Логування додавання URL до чорного списку
        log_event("Додавання URL до чорного списку: " + url)
        self.blacklist.add(url)
        self.save_blacklist()

    def add_url_to_whitelist(self, url):
        #Логування додавання URL до білого списку
        log_event("Додавання URL до білого списку: " + url)
        self.whitelist.add(url)
        self.save_whitelist()

#Функція для проведення детального аналізу контенту
def perform_detailed_analysis(content):
    #Логування початку детального аналізу контенту
    log_event("Початок детального аналізу контенту.")
    suspicious_count = 0
```

```
    suspicious_words = ["fraud", "scam", "phishing", "malware", "attack",
"danger", "threat"]
    words = content.split()
    for word in words:
        #Логування аналізу слова
        log_event("Аналіз слова: " + word)
        if word.lower() in suspicious_words:
            #Логування виявлення підозрілого слова
            log_event("Підозріле слово: " + word)
            suspicious_count += 1
    #Логування підрахунку підозрілих слів
    log_event("Загальна кількість підозрілих слів: " + str(suspicious_count))
    return suspicious_count

#Функція для отримання контенту веб-сторінки
def fetch_page_content(url):
    #Логування запиту на отримання контенту з URL
    log_event("Отримання контенту з URL: " + url)
    try:
        response = requests.get(url, timeout=5)
        if response.status_code == 200:
            #Логування успішного отримання контенту
            log_event("Контент отримано успішно з URL: " + url)
            return response.text
        else:
            #Логування помилки отримання контенту
            log_event("Помилка отримання контенту, статус: " +
str(response.status_code))
            return ""
    except Exception as e:
        #Логування винятку при отриманні контенту
        log_event("Виняток при отриманні контенту: " + str(e))
        return ""

#Функція для аналізу веб-сторінки на наявність шахрайства
def analyze_page_for_fraud(url):
    #Логування початку аналізу сторінки
    log_event("Аналіз сторінки на шахрайство: " + url)
    content = fetch_page_content(url)
    suspicious_score = perform_detailed_analysis(content)
    threshold = 3
    #Логування встановлення порогу підозри
    log_event("Встановлений поріг підозри: " + str(threshold))
    if suspicious_score >= threshold:
        #Логування виявлення шахрайства
        log_event("Шахрайство виявлено на сторінці: " + url)
        return True
    else:
        #Логування відсутності шахрайства
        log_event("Шахрайство не виявлено на сторінці: " + url)
        return False

#Функція для симуляції роботи плагіну браузера
def simulate_browser_plugin():
    #Логування старту симуляції плагіну
    log_event("Симуляція плагіну браузера розпочато.")
    plugin = CyberSecurityPlugin()
    test_urls = [
        "http://example.com/home",
        "http://malicious.net/offer",
        "http://testsite.org/login",
        "http://safeplace.io/contact",
        "http://phishing.com/free",
```

```
        "http://scam.net/win",
        "http://fraudulent.co/dashboard",
        "http://phishing.org/promo"
    ]
    for url in test_urls:
        #Логування обробки тестового URL
        log_event("Обробка тестового URL: " + url)
        safe = plugin.is_url_safe(url)
        if not safe:
            #Логування виклику аналізу шахрайства
            fraud_detected = analyze_page_for_fraud(url)
            if fraud_detected:
                #Логування реєстрації інциденту
                plugin.report_incident(url, "Виявлено підозрілу активність,
можлива спроба шахрайства.")
        else:
            #Логування перевірки URL, який пройшов перевірку
            log_event("URL пройшов перевірку: " + url)
        #Затримка між перевірками
        time.sleep(1)

#Функція для перевірки системних налаштувань
def system_configuration_check():
    #Логування початку перевірки системних налаштувань
    log_event("Перевірка системних налаштувань розпочато.")
    python_version = sys.version
    #Логування версії Python
    log_event("Версія Python: " + python_version)
    network_status = "Connected"
    #Логування статусу мережі
    log_event("Статус мережі: " + network_status)
    return True

#Функція для оновлення бази даних чорного списку
def update_blacklist_database():
    #Логування оновлення бази даних чорного списку
    log_event("Оновлення бази даних чорного списку розпочато.")
    try:
        response = requests.get("https://api.example.com/new_blacklist",
timeout=5)
        if response.status_code == 200:
            data = response.json()
            #Логування успішного отримання нових записів
            log_event("Нові записи чорного списку отримано успішно.")
            return data.get('blacklist', [])
        else:
            #Логування помилки оновлення бази даних
            log_event("Помилка оновлення бази даних, статус: " +
str(response.status_code))
            return []
    except Exception as e:
        #Логування винятку при оновленні бази даних
        log_event("Виняток при оновленні бази даних чорного списку: " + str(e))
        return []

#Функція для інтеграції оновлень чорного списку в плагін
def integrate_blacklist_updates(plugin):
    #Логування інтеграції оновлень чорного списку
    log_event("Інтеграція оновлень чорного списку розпочато.")
    new_entries = update_blacklist_database()
    for url in new_entries:
        #Логування інтеграції нового URL до чорного списку
        log_event("Інтеграція нового URL до чорного списку: " + url)
```

```
        plugin.add_url_to_blacklist(url)
        #Невелика затримка між інтеграціями
        time.sleep(0.5)

#Функція для виконання додаткового сканування безпеки
def dummy_security_scan():
    #Логування старту додаткового сканування системи
    log_event("Додаткове сканування системи розпочато.")
    for i in range(5):
        #Логування кроку додаткового сканування
        log_event("Додаткове сканування, крок: " + str(i+1))
        time.sleep(0.3)
    #Логування завершення додаткового сканування
    log_event("Додаткове сканування системи завершено.")

#Функція для запуску діагностики системи
def run_diagnostics():
    #Логування запуску діагностики системи
    log_event("Запуск діагностики системи розпочато.")
    dummy_security_scan()
    #Логування завершення діагностики системи
    log_event("Діагностика системи завершена.")
    return True

#Функція для генерації звіту по стану системи
def generate_report():
    #Логування генерації звіту
    log_event("Генерація звіту по стану системи розпочато.")
    report = {}
    report['timestamp'] = time.time()
    report['api_query_count'] = "Лічильник запитів: " + str(random.randint(10,
100))
    report['blacklist_size'] = random.randint(50, 150)
    report['whitelist_size'] = random.randint(200, 500)
    #Логування завершення генерації звіту
    log_event("Звіт згенеровано успішно.")
    return report

#Функція для виконання додаткової функціональності
def extra_functionality():
    #Логування старту додаткової функціональності
    log_event("Старт додаткової функціональності розпочато.")
    for j in range(3):
        #Логування ітерації додаткової функціональності
        log_event("Додаткова функціональність, ітерація: " + str(j+1))
        time.sleep(0.4)
    #Логування завершення додаткової функціональності
    log_event("Додаткова функціональність завершена.")

#Функція для запуску розширених функцій плагіну
def extended_plugin_features():
    #Логування запуску розширених функцій плагіну
    log_event("Запуск розширених функцій плагіну розпочато.")
    extra_functionality()
    run_diagnostics()
    report = generate_report()
    #Логування виведення звіту по розширених функціях
    log_event("Розширені функції плагіну завершено. Звіт: " + str(report))
    return report

#Функція для виконання тестового набору
def test_suite():
    #Логування запуску тестового набору
```

```
    log_event("Запуск тестового набору розпочато.")
    plugin = CyberSecurityPlugin()
    test_urls = [
        "http://example.com/test",
        "http://scam.net/test",
        "http://phishing.org/test",
        "http://safeplace.io/test",
        "http://fraudulent.co/test"
    ]
    for url in test_urls:
        #Логування тестування URL
        log_event("Тестування URL: " + url)
        safe = plugin.is_url_safe(url)
        if not safe:
            #Логування виявлення небезпечного URL у тесті
            log_event("Тест: небезпечний URL виявлено: " + url)
        else:
            #Логування успішного проходження тесту для URL
            log_event("Тест: URL пройшов перевірку: " + url)
        time.sleep(1)
    #Логування завершення тестового набору
    log_event("Тестовий набір завершено.")

#Основна функція запуску програми
def main():
    #Логування старту основного модуля плагіну
    log_event("Старт основного модуля плагіну розпочато.")
    if system_configuration_check():
        #Логування успішної перевірки системних налаштувань
        log_event("Системні налаштування перевірено успішно.")
    else:
        #Логування невдалої перевірки системних налаштувань
        log_event("Помилка перевірки системних налаштувань.")
    simulate_browser_plugin()
    plugin = CyberSecurityPlugin()
    integrate_blacklist_updates(plugin)
    extended_plugin_features()
    test_suite()
    #Затримка перед завершенням роботи плагіну
    time.sleep(5)
    plugin.stop_monitoring()
    #Логування завершення роботи плагіну
    log_event("Основний модуль плагіну завершив роботу.")

if __name__ == '__main__':
    #Логування входу в основний модуль програми
    log_event("Вхід в основний модуль програми.")
    main()
```

Файл URLAnalyzerML.py

```
import threading
import time
import random
import requests
import json
import datetime
import os
import numpy as np
from flask import Flask, jsonify, request
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer

class URLAnalyzerML:
    def __init__(self):
        self.vectorizer = CountVectorizer(analyzer='char', ngram_range=(2,4))
        self.model = LogisticRegression()
        self.train_model()

    def train_model(self):
        urls = ["http://safe.com", "http://malicious.com",
"http://goodsite.org", "http://badsite.net", "http://example.com",
"http://phishingsite.com", "http://legit.org", "http://scam.com"]
        labels = [0, 1, 0, 1, 0, 1, 0, 1]
        features = self.vectorizer.fit_transform(urls)
        self.model.fit(features, labels)

    def extract_features(self, url):
        return self.vectorizer.transform([url])

    def predict(self, url):
        features = self.extract_features(url)
        prediction = self.model.predict(features)
        return prediction[0]

    def classify_url(self, url):
        result = self.predict(url)
        if result == 1:
            return False
        else:
            return True

class UserBehaviorAnalytics:
    def __init__(self):
        self.events = []

    def record_event(self, user_id, event_type, details, timestamp):
        event = {"user_id": user_id, "event_type": event_type, "details":
details, "timestamp": timestamp}
        self.events.append(event)

    def get_user_stats(self, user_id):
        user_events = [event for event in self.events if event["user_id"] ==
user_id]
        count = len(user_events)
        types = {}
        for event in user_events:
            if event["event_type"] in types:
                types[event["event_type"]] += 1
            else:
                types[event["event_type"]] = 1
        return {"total_events": count, "event_breakdown": types}
```

```
    def get_global_stats(self):
        count = len(self.events)
        users = {}
        for event in self.events:
            uid = event["user_id"]
            if uid in users:
                users[uid] += 1
            else:
                users[uid] = 1
        return {"total_events": count, "user_activity": users}

    def clear_events(self):
        self.events = []

    def simulate_event_generation(self, num_events=50):
        user_ids = ["user1", "user2", "user3", "user4", "user5"]
        event_types = ["click", "scroll", "input", "navigation", "download"]
        for i in range(num_events):
            uid = random.choice(user_ids)
            etype = random.choice(event_types)
            details = "Detail_" + str(random.randint(1, 100))
            ts = datetime.datetime.now().isoformat()
            self.record_event(uid, etype, details, ts)
            time.sleep(0.1)

class ReportingPanel:
    def __init__(self, behavior_analytics, audit_logger, threat_updater,
url_analyzer):
        self.app = Flask(__name__)
        self.behavior_analytics = behavior_analytics
        self.audit_logger = audit_logger
        self.threat_updater = threat_updater
        self.url_analyzer = url_analyzer
        self.setup_routes()

    def setup_routes(self):
        @self.app.route("/")
        def index():
            report = self.generate_report()
            html = "<html><head><title>Security
Report</title></head><body><h1>Security Report</h1><pre>" + json.dumps(report,
indent=4) + "</pre></body></html>"
            return html
        @self.app.route("/api/report")
        def api_report():
            report = self.generate_report()
            return jsonify(report)
        @self.app.route("/api/analyze", methods=["POST"])
        def analyze():
            data = request.get_json()
            url = data.get("url", "")
            classification = self.url_analyzer.classify_url(url)
            result = {"url": url, "safe": classification}
            return jsonify(result)

    def generate_report(self):
        user_stats = self.behavior_analytics.get_global_stats()
        audit_logs = self.audit_logger.get_logs()
        threats = self.threat_updater.get_threats()
        report = {"timestamp": datetime.datetime.now().isoformat(),
"user_stats": user_stats, "audit_logs": audit_logs, "threats": threats}
        return report
```

```
    def run_server(self, host="127.0.0.1", port=5000):
        self.app.run(host=host, port=port)

class ThreatDatabaseUpdater:
    def __init__(self, update_interval=10):
        self.threats = []
        self.update_interval = update_interval
        self.running = True
        self.thread = threading.Thread(target=self.update_loop)
        self.thread.start()

    def update_loop(self):
        while self.running:
            self.update_database()
            time.sleep(self.update_interval)

    def update_database(self):
        try:
            response = requests.get("https://example.com/threats")
            if response.status_code == 200:
                data = response.json()
                self.threats = data.get("threats", [])
            else:
                self.threats = []
        except Exception as e:
            self.threats = []

    def get_threats(self):
        return self.threats

    def stop_updates(self):
        self.running = False
        self.thread.join()

class AuditLogger:
    def __init__(self):
        self.logs = []

    def log_event(self, event_type, details, timestamp):
        log_entry = {"event_type": event_type, "details": details, "timestamp":
timestamp}
        self.logs.append(log_entry)

    def get_logs(self):
        return self.logs

    def clear_logs(self):
        self.logs = []

    def save_logs_to_file(self, file_path):
        with open(file_path, "w") as f:
            json.dump(self.logs, f, indent=4)

    def load_logs_from_file(self, file_path):
        if os.path.exists(file_path):
            with open(file_path, "r") as f:
                self.logs = json.load(f)
        else:
            self.logs = []

    def filter_logs_by_type(self, event_type):
        return [log for log in self.logs if log["event_type"] == event_type]
```

```
def main():
    url_analyzer = URLAnalyzerML()
    behavior_analytics = UserBehaviorAnalytics()
    audit_logger = AuditLogger()
    threat_updater = ThreatDatabaseUpdater(update_interval=15)
    reporting_panel = ReportingPanel(behavior_analytics, audit_logger,
threat_updater, url_analyzer)
    def run_flask():
        reporting_panel.run_server()
    flask_thread = threading.Thread(target=run_flask)
    flask_thread.daemon = True
    flask_thread.start()
    for i in range(20):
        url = "http://testsite" + str(i) + ".com/path" + str(i)
        safe = url_analyzer.classify_url(url)
        event_type = "URL_Safe" if safe else "URL_Unsafe"
        audit_logger.log_event(event_type, url,
datetime.datetime.now().isoformat())
        behavior_analytics.record_event("user" + str(i % 5), event_type, url,
datetime.datetime.now().isoformat())
        time.sleep(0.2)
    behavior_analytics.simulate_event_generation(30)
    audit_logger.save_logs_to_file("audit_logs.json")
    threat_data = threat_updater.get_threats()
    time.sleep(5)
    threat_updater.stop_updates()

if __name__ == "__main__":
    main()
```

Файл BrowserHistoryScanner.py

```
import os
import re
import time
import random
import json
import datetime
import requests
import threading
import base64
import hmac
import hashlib

class BrowserHistoryScanner:
    def __init__(self, history_file='browser_history.json'):
        self.history_file = history_file
        self.history = []
        self.suspicious_urls = []
    def load_history(self):
        if os.path.exists(self.history_file):
            with open(self.history_file, 'r', encoding='utf-8') as f:
                try:
                    self.history = json.load(f)
                except Exception as e:
                    self.history = []
        else:
            self.history = []
    def parse_history(self):
        urls = []
        for entry in self.history:
            if isinstance(entry, dict) and 'url' in entry:
                urls.append(entry['url'])
            elif isinstance(entry, str):
                urls.append(entry)
        return urls
    def analyze_history(self):
        urls = self.parse_history()
        pattern = re.compile(r'(login|account|verify|update)', re.IGNORECASE)
        for url in urls:
            if pattern.search(url):
                self.suspicious_urls.append(url)
        return self.suspicious_urls
    def generate_history_report(self):
        report = {}
        report['total_entries'] = len(self.history)
        report['suspicious_count'] = len(self.suspicious_urls)
        report['suspicious_urls'] = self.suspicious_urls
        return report

class AntivirusIntegration:
    def __init__(self):
        self.scan_results = {}
    def scan_file(self, file_path):
        time.sleep(0.2)
        risk_score = random.randint(0, 100)
        if risk_score > 70:
            result = 'infected'
        else:
            result = 'clean'
        self.scan_results[file_path] = {'risk_score': risk_score, 'result':
result}
        return self.scan_results[file_path]
```

```
    def get_scan_result(self, file_path):
        if file_path in self.scan_results:
            return self.scan_results[file_path]
        else:
            return self.scan_file(file_path)
    def simulate_virus_scan(self, files):
        results = {}
        for f in files:
            result = self.scan_file(f)
            results[f] = result
            time.sleep(0.1)
        return results

class GeolocationThreatAnalyzer:
    def __init__(self):
        self.api_url = "https://ipapi.co/"
        self.threat_regions = ['CN', 'RU', 'KP']
        self.lookup_results = {}
    def lookup_ip(self, ip):
        try:
            response = requests.get(self.api_url + ip + "/json/", timeout=3)
            if response.status_code == 200:
                data = response.json()
                self.lookup_results[ip] = data
                return data
            else:
                self.lookup_results[ip] = {}
                return {}
        except Exception as e:
            self.lookup_results[ip] = {}
            return {}
    def analyze_geolocation(self, ip):
        data = self.lookup_ip(ip)
        if 'country' in data and data['country'] in self.threat_regions:
            return True
        return False
    def generate_geolocation_report(self, ip_list):
        report = {}
        for ip in ip_list:
            report[ip] = {'data': self.lookup_ip(ip), 'threat':
self.analyze_geolocation(ip)}
            time.sleep(0.1)
        return report

class PhishingEmailFilter:
    def __init__(self):
        self.phishing_patterns = [r'urgent action required', r'account
suspended', r'verify your account', r'click here immediately', r'update your
information']
        self.quarantined_emails = []
    def filter_email(self, email_content):
        for pattern in self.phishing_patterns:
            if re.search(pattern, email_content, re.IGNORECASE):
                return True
        return False
    def extract_links(self, email_content):
        links = re.findall(r'(https?://[^\s]+)', email_content)
        return links
    def analyze_email(self, email_content):
        result = {}
        result['is_phishing'] = self.filter_email(email_content)
        result['links'] = self.extract_links(email_content)
        result['length'] = len(email_content)
```

```
        return result
    def quarantine_email(self, email_id, email_content):
        analysis = self.analyze_email(email_content)
        if analysis['is_phishing']:
            self.quarantined_emails.append({'email_id': email_id, 'content':
email_content, 'analysis': analysis})
            return True
        return False
    def get_quarantined_emails(self):
        return self.quarantined_emails

class TwoFactorAuthentication:
    def __init__(self, secret=None):
        if secret is None:
            self.secret = base64.b32encode(os.urandom(10)).decode('utf-8')
        else:
            self.secret = secret
    def generate_otp(self, interval=30):
        timestep = int(time.time() // interval)
        key = base64.b32decode(self.secret, True)
        msg = timestep.to_bytes(8, 'big')
        h = hmac.new(key, msg, hashlib.sha1).digest()
        o = h[19] & 15
        otp = (int.from_bytes(h[o:o+4], 'big') & 0x7fffffff) % 1000000
        return str(otp).zfill(6)
    def verify_otp(self, user_otp, interval=30):
        generated = self.generate_otp(interval)
        return user_otp == generated
    def simulate_user_authentication(self, user_input_function):
        otp = self.generate_otp()
        for attempt in range(3):
            user_otp = user_input_function()
            if self.verify_otp(user_otp):
                return True
            time.sleep(1)
        return False
    def update_secret(self):
        self.secret = base64.b32encode(os.urandom(10)).decode('utf-8')
        return self.secret

def simulate_browser_history():
    scanner = BrowserHistoryScanner()
    sample_history = [
        {"url": "http://example.com/home", "timestamp": "2021-09-01T10:00:00"},
        {"url": "http://bank.com/login", "timestamp": "2021-09-01T10:05:00"},
        {"url": "http://shop.com/account", "timestamp": "2021-09-01T10:10:00"},
        {"url": "http://news.com", "timestamp": "2021-09-01T10:15:00"},
        {"url": "http://verify.com/update", "timestamp": "2021-09-01T10:20:00"}
    ]
    with open('browser_history.json', 'w', encoding='utf-8') as f:
        json.dump(sample_history, f, indent=4)
    scanner.load_history()
    scanner.analyze_history()
    report = scanner.generate_history_report()
    return report

def simulate_antivirus_scan():
    av = AntivirusIntegration()
    sample_files = ['file1.exe', 'file2.dll', 'file3.sys', 'file4.doc',
'file5.pdf']
    results = av.simulate_virus_scan(sample_files)
    return results
```

```
def simulate_geolocation_analysis():
    geo = GeolocationThreatAnalyzer()
    ip_list = ['8.8.8.8', '1.1.1.1', '213.180.204.3', '185.220.101.1',
'91.198.174.192']
    report = geo.generate_geolocation_report(ip_list)
    return report

def simulate_phishing_email_filter():
    filter_obj = PhishingEmailFilter()
    emails = [
        {"id": "email1", "content": "Dear user, urgent action required. Please
click here immediately to verify your account."},
        {"id": "email2", "content": "Hello, your order has been shipped. Track
it here: http://shipping.com/track"},
        {"id": "email3", "content": "Important: account suspended. Update your
information at http://secure-update.com"},
        {"id": "email4", "content": "Greetings, we have a newsletter for you.
Enjoy our latest news."},
        {"id": "email5", "content": "Notice: verify your account details at
http://verify-now.com"}
    ]
    quarantine_results = {}
    for email in emails:
        result = filter_obj.quarantine_email(email["id"], email["content"])
        quarantine_results[email["id"]] = result
    return {"quarantined": filter_obj.get_quarantined_emails(), "results":
quarantine_results}

def simulate_two_factor_authentication():
    auth = TwoFactorAuthentication()
    def dummy_input():
        return auth.generate_otp()
    auth_success = auth.simulate_user_authentication(dummy_input)
    new_secret = auth.update_secret()
    return {"authentication_success": auth_success, "new_secret": new_secret}

def main():
    history_report = simulate_browser_history()
    print("Browser History Report:")
    print(json.dumps(history_report, indent=4))
    av_results = simulate_antivirus_scan()
    print("Antivirus Scan Results:")
    print(json.dumps(av_results, indent=4))
    geo_report = simulate_geolocation_analysis()
    print("Geolocation Analysis Report:")
    print(json.dumps(geo_report, indent=4))
    phishing_results = simulate_phishing_email_filter()
    print("Phishing Email Filter Results:")
    print(json.dumps(phishing_results, indent=4))
    twofa_results = simulate_two_factor_authentication()
    print("Two-Factor Authentication Simulation Results:")
    print(json.dumps(twofa_results, indent=4))

if __name__ == "__main__":
    main()
```

Файл SocialMediaAnalyzer.py

```
import time
import threading
import random
import json
import requests
import base64
import os
import hmac
import hashlib

class SocialMediaAnalyzer:
    def __init__(self):
        self.posts = []
        self.suspicious_keywords = ["fraud", "scam", "phishing", "malware",
"attack", "hoax", "fake", "cheat"]
        self.suspicious_posts = []
    def fetch_posts(self, account):
        fetched = []
        for i in range(10):
            post = {"id": f"{account}_{i}", "content": f"This is a sample post
number {i} from {account}."}
            if random.random() < 0.3:
                post["content"] += " fraud alert"
            fetched.append(post)
            time.sleep(0.05)
        self.posts.extend(fetched)
        return fetched
    def analyze_posts(self):
        self.suspicious_posts = []
        for post in self.posts:
            count = 0
            words = post["content"].split()
            for word in words:
                if word.lower() in self.suspicious_keywords:
                    count += 1
            if count >= 1:
                self.suspicious_posts.append(post)
        return self.suspicious_posts
    def get_suspicious_posts(self):
        return self.suspicious_posts
    def simulate_analysis(self, accounts):
        for account in accounts:
            self.fetch_posts(account)
        self.analyze_posts()
        return self.get_suspicious_posts()

class CustomNotificationSystem:
    def __init__(self):
        self.channels = {}
        self.notification_queue = []
        self.running = True
        self.thread = threading.Thread(target=self.process_queue)
        self.thread.start()
    def add_channel(self, channel_name, handler):
        self.channels[channel_name] = handler
    def remove_channel(self, channel_name):
        if channel_name in self.channels:
            del self.channels[channel_name]
    def send_notification(self, channel_name, message):
        if channel_name in self.channels:
            self.channels[channel_name](message)
```

```
    def schedule_notification(self, channel_name, message, delay):
        scheduled_time = time.time() + delay
        self.notification_queue.append((scheduled_time, channel_name, message))
    def process_queue(self):
        while self.running:
            now = time.time()
            for item in self.notification_queue[:]:
                scheduled_time, channel_name, message = item
                if now >= scheduled_time:
                    self.send_notification(channel_name, message)
                    self.notification_queue.remove(item)
            time.sleep(0.1)
    def stop(self):
        self.running = False
        self.thread.join()

class ExtensionMarketplace:
    def __init__(self):
        self.available_extensions = {
            "AdBlocker": {"version": "1.0", "size": 2048, "dependencies": []},
            "PrivacyGuard": {"version": "2.1", "size": 4096, "dependencies":
["AdBlocker"]},
            "SecureLogin": {"version": "1.5", "size": 1024, "dependencies": []},
            "QuickTranslate": {"version": "3.0", "size": 3072, "dependencies":
[]},
            "ThemeChanger": {"version": "1.2", "size": 512, "dependencies": []}
        }
        self.installed_extensions = {}
    def list_extensions(self):
        return self.available_extensions
    def search_extension(self, keyword):
        results = {}
        for name, info in self.available_extensions.items():
            if keyword.lower() in name.lower():
                results[name] = info
        return results
    def install_extension(self, extension_name):
        if extension_name in self.available_extensions:
            ext_info = self.available_extensions[extension_name]
            for dep in ext_info.get("dependencies", []):
                if dep not in self.installed_extensions:
                    self.install_extension(dep)
            self.installed_extensions[extension_name] = ext_info
            time.sleep(0.2)
            return True
        return False
    def uninstall_extension(self, extension_name):
        if extension_name in self.installed_extensions:
            del self.installed_extensions[extension_name]
            time.sleep(0.1)
            return True
        return False
    def update_extension(self, extension_name, new_version):
        if extension_name in self.installed_extensions:
            self.installed_extensions[extension_name]["version"] = new_version
            time.sleep(0.1)
            return True
        return False
    def get_installed_extensions(self):
        return self.installed_extensions

class DataEncryptionModule:
    def __init__(self, key=None):
```

```
            if key is None:
                self.key = self.generate_key()
            else:
                self.key = key
        def generate_key(self):
            return base64.urlsafe_b64encode(os.urandom(16)).decode('utf-8')
        def update_key(self):
            self.key = self.generate_key()
            return self.key
        def encrypt(self, plaintext):
            key_bytes = self.key.encode('utf-8')
            plain_bytes = plaintext.encode('utf-8')
            encrypted = bytearray()
            for i in range(len(plain_bytes)):
                encrypted.append(plain_bytes[i] ^ key_bytes[i % len(key_bytes)])
            return base64.urlsafe_b64encode(encrypted).decode('utf-8')
        def decrypt(self, ciphertext):
            key_bytes = self.key.encode('utf-8')
            try:
                encrypted = base64.urlsafe_b64decode(ciphertext)
            except Exception as e:
                return ""
            decrypted = bytearray()
            for i in range(len(encrypted)):
                decrypted.append(encrypted[i] ^ key_bytes[i % len(key_bytes)])
            return decrypted.decode('utf-8', errors='ignore')

    class RemoteAdministration:
        def __init__(self):
            self.remote_config = {}
            self.connected = False
            self.command_history = []
            self.response_history = []
        def connect(self, address, port):
            self.remote_config["address"] = address
            self.remote_config["port"] = port
            time.sleep(0.2)
            self.connected = True
            return self.connected
        def disconnect(self):
            self.connected = False
            time.sleep(0.1)
            return self.connected
        def send_command(self, command):
            if self.connected:
                self.command_history.append(command)
                response = self.simulate_remote_response(command)
                self.response_history.append(response)
                return response
            return None
        def simulate_remote_response(self, command):
            responses = {
                "status": "All systems operational",
                "restart": "System restarting",
                "update": "Configuration updated",
                "shutdown": "System shutting down"
            }
            return responses.get(command, "Unknown command")
        def update_configuration(self, key, value):
            if self.connected:
                self.remote_config[key] = value
                time.sleep(0.1)
                return True
```

```
            return False
        def monitor_remote_system(self, duration):
            log = []
            start_time = time.time()
            while time.time() - start_time < duration:
                status = self.send_command("status")
                log.append(status)
                time.sleep(0.5)
            return log

def simulate_social_media_analysis():
    analyzer = SocialMediaAnalyzer()
    accounts = ["accountA", "accountB", "accountC"]
    suspicious = analyzer.simulate_analysis(accounts)
    print("Suspicious Social Media Posts:")
    print(json.dumps(suspicious, indent=4))

def simulate_notification_system():
    def email_handler(message):
        print("Email Notification:", message)
    def sms_handler(message):
        print("SMS Notification:", message)
    notifier = CustomNotificationSystem()
    notifier.add_channel("email", email_handler)
    notifier.add_channel("sms", sms_handler)
    notifier.schedule_notification("email", "Alert: Suspicious activity
detected", 1)
    notifier.schedule_notification("sms", "Alert: Suspicious activity detected",
2)
    time.sleep(3)
    notifier.stop()

def simulate_extension_marketplace():
    marketplace = ExtensionMarketplace()
    print("Available Extensions:")
    print(json.dumps(marketplace.list_extensions(), indent=4))
    marketplace.install_extension("PrivacyGuard")
    marketplace.install_extension("ThemeChanger")
    print("Installed Extensions:")
    print(json.dumps(marketplace.get_installed_extensions(), indent=4))
    marketplace.update_extension("ThemeChanger", "1.3")
    print("Updated Extensions:")
    print(json.dumps(marketplace.get_installed_extensions(), indent=4))
    marketplace.uninstall_extension("PrivacyGuard")
    print("After Uninstallation:")
    print(json.dumps(marketplace.get_installed_extensions(), indent=4))

def simulate_data_encryption():
    encryption = DataEncryptionModule()
    plaintext = "Sensitive data that needs encryption"
    ciphertext = encryption.encrypt(plaintext)
    decrypted = encryption.decrypt(ciphertext)
    print("Original:", plaintext)
    print("Encrypted:", ciphertext)
    print("Decrypted:", decrypted)
    new_key = encryption.update_key()
    ciphertext2 = encryption.encrypt(plaintext)
    decrypted2 = encryption.decrypt(ciphertext2)
    print("New Key:", new_key)
    print("Encrypted with New Key:", ciphertext2)
    print("Decrypted with New Key:", decrypted2)

def simulate_remote_administration():
```

```
    remote = RemoteAdministration()
    remote.connect("192.168.1.100", 8080)
    status_response = remote.send_command("status")
    print("Status Response:", status_response)
    remote.update_configuration("mode", "maintenance")
    log = remote.monitor_remote_system(3)
    print("Monitoring Log:")
    print(json.dumps(log, indent=4))
    remote.disconnect()

def main():
    simulate_social_media_analysis()
    simulate_notification_system()
    simulate_extension_marketplace()
    simulate_data_encryption()
    simulate_remote_administration()

if __name__ == "__main__":
    main()
```