МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р. техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2023 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ВИЯВЛЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ ТА МЕТОДИ ЇХ УСУНЕННЯ

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21910220

Виконав студент 4-го курсу, групи 402
_____ І. О. Пустовіт
«20» червня 2023 р.

Керівник: канд. фіз.-мат. наук, доцент
_____І. В. Кулаковська
«20» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р. техн. наук, проф.
_____ Ю. П. Кондратенко
«23» листопада 2022 р.

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

Видано студенту групи 402 факультету комп'ютерних наук Пустовіту Ігорю Олеговичу.

1. Тема кваліфікаційної роботи «Виявлення вразливостей вебзастосунків та методи їх усунення».

Керівник роботи Кулаковська Інесса Василівна, канд. фіз.-мат. наук, доцент

Затв. наказом Ректора ЧНУ ім. Петра Могили від «17» березня 2023 р. № 59

2. Строк представлення кваліфікаційної роботи студентом «20» червня 2023 р.

3. Вхідні (початкові) дані до роботи: тестове середовище у вигляді вебзастосунку, сканери вразливостей безпеки вебзастосунків, інформація у мережі Інтернет та наукова література.

Очікуваний результат: готове дослідження сканерів вразливостей безпеки вебзастосунків, рекомендаційна система для проведення тестувань безпеки та усунення вразливостей

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

– дослідження методологій тестування на проникнення;

– дослідження проблеми кібератак в Україні до та після початку повномасштабного вторгнення;

– дослідження моделі Take-Grant;

– порівняльний аналіз сканерів вразливостей безпеки вебзастосунків;

– рекомендаційна система для проведення тестувань безпеки та усунення вразливостей.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: «Вимоги щодо використання екранних пристроїв».

7. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Боженко А. Л., викладач	

Керівник роботи __канд. фіз.-мат._наук_І. В. Кулаковська_____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Пустовіт І. О._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_23_» _____листопада____ 2022 р.

КАЛЕНДАРНИЙ ПЛАН
виконання бакалавської кваліфікаційної роботи

Тема: Виявлення вразливостей вебзастосунків та методи їх усунення

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників БКР	26.10.2022	26.10.2022	Виконано
2	Отримання завдання на виконання БКР	23.11.2022	23.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	05.12.2022	05.12.2022	Виконано
4	Отримання завдання на переддипломну практику	20.04.2023	20.04.2023	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до БКР	01.05.2023	14.05.2022	Виконано
6	Розробка звіту з переддипломної практики	15.05.2023	17.05.2023	Виконано
7	Виконання БКР: дослідження класифікацій вразливостей безпеки та методологій тестування безпеки, дослідження проблеми кібератак в Україні, модель Take-Grant, порівняльний аналіз сканерів вразливостей безпеки, рекомендаційна система	15.05.2023	19.06.2023	Виконано
8	Попередній захист БКР на засіданні комісії кафедри	29.05.2023	30.05.2023	Виконано
9	Доробка та остаточне оформлення БКР	02.06.2023	19.06.2023	Виконано
10	Подання БКР рецензенту	15.06.2023	17.06.2023	Виконано
11	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.06.2023	22.06.2023	
12	Захист БКР перед екзаменаційною комісією (ЕК)	26.06.2023	29.06.2023	

Розробив студент: _____Пустовіт І. О._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи канд. фіз.-мат. наук Кулаковська І. В.____ _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«23» листопада 2022 р.

АНОТАЦІЯ

бакалаврської кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили

Пустовіта Ігоря Олеговича

Тема: «Виявлення вразливостей вебзастосунків та методи їх усунення»

Актуальність**:** практика проведення аудитів безпеки вебзастосунків сприяла б плануванню, створенню та адмініструванню безпечних програмних продуктів, що допомогло б користувачам вебзастосунків використовувати програмний продукт без ризику стати жертвами хакерської атаки, внаслідок якої можливо втратити доступ до важливої інформації, стати жертвою витоку такої інформації або отримати значні фінансові збитки.

Об’єкт роботи: процес тестування вебзастосунку на проникнення та створення рекомендаційної системи для спрощення проведення тестувань безпеки та усунення вразливостей.

Предмет роботи: сканери вразливостей безпеки вебзастосунків, інформація про методології тестування на проникнення, актуальність проблеми кібератак в Україні, модель Take-Grant та методи усунення вразливостей безпеки.

Мета роботи: виявлення вразливостей безпеки у вебзастосунках задля демонстрації важливості створення, конфігурування та адміністрування безпечних вебзастосунків; створення рекомендаційної системи.

У першому розділі проводиться дослідження необхідності проведення тестувань на проникнення та методологій проведення тестувань. У другому розділі проводиться дослідження моделі розмежування доступу Take-Grant та дослідження актуальності проблеми кібератак в Україні. У третьому розділі проводиться порівняльний аналіз сканерів вразливостей безпеки. У четвертому розділі описується рекомендаційна система для проведення тестувань на проникнення із методами для усунення вразливостей безпеки.

Бакалаврська кваліфікаційна робота містить 91 сторінку, 1 таблицю, 41 рисунок, 30 посилань, 2 додатки.

Ключові слова: вебзастосунок, тестове середовище, вразливість, кібератака, аудит безпеки, методології тестування на проникнення, сканер, рекомендаційна система, модель розмежування доступу, мова програмування C#.

ABSTRACT

for bachelor's qualification work of a student of 402 group at Petro Mohyla Black Sea National University

Pustovit Ihor Olehovych

Topic: «Identification of web application vulnerabilities and methods of their mitigation»

Relevance: the practice of conducting security audits of web applications would facilitate the planning, creation and administration of secure software products, which would help users of web applications use the software product without the risk of falling victim to a hacker attack, which could result in losing access to important information, becoming a victim of leakage of such information or suffering significant financial losses.

The object of the work is the process of testing a web application for penetration and the process of creating a recommendation system using the C# programming language for conducting security tests and mitigating vulnerabilities.

The subject of the work is web application security vulnerability scanners, information on penetration testing methodologies, researches on the problem of cyberattacks in Ukraine, and methods of mitigating web application security vulnerabilities.

The purpose of the work is identifying security vulnerabilities in web applications to demonstrate the importance of creating, configuring, and administering secure web applications; creating a recommendation system.

The first section analyzes the need for penetration testing and the methodologies for conducting tests. The second section analyzes the Take-Grant access control model and studies the relevance of the problem of cyberattacks in Ukraine. The third section provides a comparative analysis of security vulnerability scanners. The fourth section describes a recommendation system for conducting penetration tests with methods for mitigating security vulnerabilities.

The bachelor's thesis contains 91 pages, 1 tables, 41 pictures, 30 references, 2 appendices.

Keywords: web application, test environment, vulnerability, cyberattack, security audit, penetration testing methodologies, scanner, recommendation system, access control model, programming language C#.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних
ІБ – інформаційна безпека
ІС – інформаційна система
ІТ – інформаційні технології
ПЗ – програмне забезпечення
ПК – персональний комп’ютер

API – application programming interface
BAC – broken access control
CORS – cross-origin resource sharing
CSP – Content Security Policy
CSRF – cross-side request forgery
CSTI – client-side template injection
XSS – cross-site scripting
DOM – document object model
DAST – dynamic application security testing
XML – external markup language
HTTP – hypertext transfer protocol
IP – internet protocol
SSL – secure sockets layer
SSRF – server-side request forgery
SQL – structured query language
TLS – transport layer security
URL – uniform resource locator
WAF – web application firewall
XXE – XML external entities

ВСТУП

Під час написання бакалаврської кваліфікаційної роботи було проаналізовано та досліджено певні актуальні питання та проблеми.

Нові сучасні вразливості інформаційних систем фахівцями кібербезпеки виявляються щодня, як і способи реалізації атак, які стають доступними для зловмисників. Для отримання впевненості у захищеності даних та програмних продуктів компанії необхідно своєчасно визначати прогалини безпеки, які вимагають пильної уваги досвідчених фахівців у галузі кібербезпеки. Тестування на проникнення є досить ефективним та наочним методом для оцінювання рівня захищеності комп'ютерної системи, наприклад, вебзастосунку. Також за допомогою пентесту можна виявити допущені програмні й технічні помилки, які часто можуть виникати при роботі.

Мета роботи полягає у виявленні вразливостей безпеки у вебзастосунках задля демонстрації важливості створення, конфігурування та адміністрування безпечних вебзастосунків та створенні рекомендаційної системи для спрощення процесу аудиту безпеки та усуненню вразливостей, знайдених внаслідок такого аудиту.

Об’єкт роботи: процес тестування вебзастосунку на проникнення за допомогою таких програмних засобів як Burp Suite Professional та Acunetix Web Vulnerability Scanner та створення рекомендаційної системи на мові програмування C# для проведення тестувань безпеки та усунення вразливостей.

Предмет роботи: сканери вразливостей безпеки вебзастосунків, інформація про методології тестування на проникнення, дослідження про актуальність проблеми кібератак в Україні, модель Take-Grant та методи усунення вразливостей безпеки вебзастосунків.

Захист вебзастосунків від зловмисників залежить від технологій і компонентів, що використовуються при їх створенні, а також від можливих вразливостей цих компонентів. Причиною виникнення вразливостей є помилки в розробці, реалізації, конфігурації та застосуванні компонентів вебзастосунків,

звідси виникає необхідність пошуку вразливостей і реагування на інформацію про них. Широкий спектр інструментів дозволяє здійснювати пошук вразливостей, але ефективність їх використання залежить від алгоритму дій, які необхідно здійснити цим пошуком. Алгоритми дій можуть бути представлені у вигляді таких спеціальних методів, що охоплюють широке коло питань кібербезпеки, тому потрібен додатковий час для аналізу існуючих методів і вибору таких компонентів. Отже, існує потреба в дослідженні сканерів безпеки вебзастосунків, яке б враховувало міжнародні досягнення у тестуванні вебзастосунків і містила б перелік можливих інструментів тестування, а також у створення рекомендаційної системи, яка б сприяла спрощенню процесу аудиту безпеки та усуненню вразливостей, знайдених внаслідок такого аудиту.

Задля досягнення поставлених цілей було використано інструменти аудиту безпеки Burp Suite та Acunetix Web Vulnerability Scanner, а також інструменти десктопної розробки, такі як мова програмування C# та фреймворк Windows Forms.

Завдання роботи:

– дослідження методологій тестування на проникнення;

– дослідження проблеми кібератак в Україні до та після початку повномасштабного вторгнення;

– дослідження моделі Take-Grant;

– порівняльний аналіз сканерів вразливостей безпеки вебзастосунків;

– рекомендаційна система для проведення тестувань безпеки та усунення вразливостей.

1 АНАЛІЗ МЕТОДОЛОГІЙ ТЕСТУВАННЯ НА ПРОНИКНЕННЯ

1.1 Тест на проникнення як метод захисту інформаційної системи

Тест на проникнення (пентест, тестування на проникнення, від англ. «pentest», «pentesting», «penetration test», «penetration testing») – основна послуга етичних хакерів. Мета такого тестування – знайти вразливості безпеки, а також проаналізувати технічну захищеність, стійкість бізнес-процесів і співробітників до хакерських атак. Результатом проведення таких робіт є багаторівневий звіт з аналізом захищеності бізнесу відносно кіберризиків і рекомендаціями щодо усунення загроз і вразливостей. Тест на проникнення дозволяє отримати об'єктивну оцінку того, наскільки легко отримати доступ до ресурсів, наприклад, корпоративної мережі або сайту якоїсь компанії, яким способом можна отримати несанкціонований доступ та через які саме вразливості безпеки це може відбутися.

Основна задача тесту на проникнення - повністю імітуючи дії зловмисника, здійснити атаку на вебсервер, сервер застосунків або баз даних, систему «інтернету речей», корпоративну мережу тощо. Тестування на проникнення може проводитися як у форматі аудиту на відповідність стандартам, так і у вигляді самостійної роботи із більш простими критеріями оцінки стану об’єкта тестування.

За своєю суттю, пентест виглядає як генеральна репетиція дій хакерів. Сьогодні ця послуга стає все популярнішою і критично необхідною в області інформаційної безпеки в усьому світі. Сенс подібних робіт полягає в санкціонованій спробі обійти існуючий комплекс засобів захисту інформаційної системи. Під час тестування аудитор грає роль зловмисника, перед яким стоїть завдання порушити інформаційну безпеку мережі або застосунку замовника. Такого тестувальника також часто називають етичним хакером через те, що під час цього тестування він повинен дотримуватись заздалегідь заключеної угоди про здійснення пентесту. Цю угоду пентестер і замовник повинні підписати

заздалегідь, щоб не допустити витоку інформації про стан захисту інформаційних систем на конкретному підприємстві. Об'єктами тестів на проникнення є різноманітні компоненти інформаційної інфраструктури: активне мережеве обладнання, сервери, робочі станції, сервіси, інформаційні системи, бази даних. Завдання пентестера – виявити в них вразливості і з'ясувати можливість їх експлуатації, після чого ліквідувати ці вразливості.

Тести на проникнення необхідно проводити регулярно, оскільки постійно з'являються нові вразливості, розробляються нові експлоїти, змінюється інфраструктура та умови, в яких функціонують інформаційні системи. У межах етичного хакінгу аудитори здійснюють повний аналіз всіх деталей досліджуваного об'єкта, вибирають відповідні сценарії атак, враховуючи людський фактор, можливо, розробляють унікальне для кожного конкретного випадку програмне забезпечення чи скрипти для спроби проникнення до інформаційної системи [1].

Під час тестування на проникнення зазвичай йдеться про усі можливі варіації знаходження патернів поведінки у об’єкті тестування та його оточенні, які можуть бути змінені таким чином, що пентестер бачитиме, як об’єкт тестування «відповідає» на такі зміни. Мета полягає в тому, щоб гарантувати, що об’єкт тестування працює надійно і безпечно при навіть необґрунтованих та нетривіальних сценаріях використання.

З точки зору ризиків безпеки, дані, що вводяться користувачем, внутрішні дані об’єкту тестування та логіка роботи об’єкту тестування зазвичай є основними місцями, де спроби впливу на об’єкт тестування можуть виявити проблеми безпеки, які зазвичай стосуються файлів, системних ресурсів, локальних або мережевих ресурсів тощо. Будь-який з цих варіантів може стати відправною точкою, вектором кібератаки.

Перед початком тестування між пентестером та замовником обговорюється режим тестування на основі рівня початкових знань виконавця про об’єкт тестування (Black Box або White Box) і рівня інформованості замовника про

тестування (Black Hat або White Hat). При виборі рівня Black Box виконавцю відомий лише діапазон зовнішніх IP-адрес об’єкту тестування. Даний підхід максимально наближений до дій хакера, дані про тестований об'єкт будуть збиратися за допомогою відкритих джерел, соціальної інженерії тощо.

У режимі White Box доступна виконавцю інформація є значно ширшою. У цій ситуації фахівцям можуть бути надані документація, вихідний код застосунку, структура мережі, а також повний доступ до об'єкту, який тестується. У режимі Black Hat про проведення робіт знають тільки керівники служби ІБ. У такому випадку, вдається перевірити рівень оперативної готовності до атак у мережевих адміністраторів та адміністраторів ІБ. У режимі White Hat виконавці працюють в постійному контакті зі службою ІБ замовника, ІТ-фахівці замовника не перешкоджають виконанню необхідних тестів, не вносять зміни у код та налаштування об’єктів тестування. Незалежно від обраних підходів до тестування, основне завдання зводиться до виявлення можливих вразливостей і оцінки ризику проникнення в систему. Відносні переваги цих підходів обговорюються.

Тестування методом Black Box імітує атаку того, хто не знайомий з системою. Тестування методом White Box моделює, що може статися під час проникнення «зсередини», або після витоку конфіденційної інформації, коли зловмисник має доступ до вихідного коду, схем побудови мереж тощо.

При проведенні тесту на проникнення важливо чітко регламентувати дії сторін, виділити узгоджені тимчасові інтервали для проведення активних дій, визначити етапність, ті чи інші обмеження, погоджувати дії під час переходу від етапу до етапу. Без виконання зазначених умов кратно зростає ризик порушення платіжних та інформаційних технологічних процесів, сервісів, реалізованих на базі об'єктів інформаційної інфраструктури, які піддаються дослідженню і т.д. Крім того, необхідно послідовно документувати отримані результати і на їх основі формувати пропозиції щодо виправлення виявлених проблем. Адже

проведення тесту не є самоціллю – важливо надалі доопрацювати результати тесту і усунути виявлені вразливості.

Особливу увагу в методології проведення тестів на проникнення звертають на оцінку виявлених вразливостей і ранжуванню їх за ступенем критичності. Класичними вразливостями зазвичай є слабкі паролі, неоновлення операційних систем чи прикладного програмного забезпечення, помилки у програмуванні прикладних систем, особливо веборієнтованих, відкриті без необхідності порти на мережевих пристроях і т.д.

Послуги, пропоновані фірмами, що надають послуги тестування на проникнення, охоплюють діапазон перевірок, починаючи від простого сканування IP-адресного простору організації на відкриті порти з метою знайти вразливі сервіси та закінчуючи повною перевіркою вихідного коду застосунку. Розробка методів тестування стосується таких об'єктів як інформаційні системи, бізнес-процеси, застосунки (мобільні, веб і десктопні), бізнес-логіка інформаційних систем. Одним із методів аудиту систем та компонентів є DAST.

DAST (Dynamic Application Security Testing) — методика тестування у режимі «чорної скриньки», в процесі використання якої можна виявляти вразливості та слабкі місця у об’єктах тестування, зазвичай вебзастосунках. Нерідко це досягається за рахунок використання методів впровадження завідомо помилкових даних у застосунку для виявлення поширених вразливостей безпеки, наприклад, ін'єкцій та міжсайтових сценаріїв [2].

1.2 Існуючі методології для проведення тестування на проникнення

На даний час найбільш розповсюдженими методологіями проведення тестування на проникнення є:

– The Open Source Security Testing Methodology Manual (OSSTMM);

– The National Institute of Standards and Technology (NIST) Special Publication 800-115;

– OWASP Testing Guide;

– Penetration Testing Execution Standard (PTES);

– MITRE ATT&CK;

– Information Systems Security Assessment Framework (ISSAF).

Комплексне рішення безпеки для застосунку повинно містити більше, ніж просто структуру, програмне забезпечення, контрольний список або інструментарій - це потребує методології. Методологія безпеки – це бекенд процесів або рішень, який визначає, що або хто проходить тестування безпеки, а також коли і що саме це тестування охоплюватиме.

Тестування є складним процесом і необхідно звести його до елементарного: достатньо зрозуміти складові цих процесів імплементації безпеки. Методологія повинна містити певні показники, щоб переконатися, що методологія була розроблена правильно, та усвідомити чи оцінити результат застосування методології в процесі імплементації best practices з кібербезпеки у вебзастосунку. Виходячи з усього вищеперерахованого, методологія тестування безпеки є обов'язковим компонентом у процесах проведення тестувань на проникнення.

1.3 Дослідження існуючих методологій для тестування на проникнення

В даному підрозділі буде розглянуто детальніше кожну з методологій. Всі вони мають багато схожого, але є і суттєві відмінності, які можуть завадити застосуванню принципів імплементації безпеки, описаних у даних методологіях.

1.3.1 OSSTMM

OSSTMM (Open Source Security Testing Methodology Manual) являє собою рецензовану методологію проведення тестів на проникнення. В OSSTMM випадки тестів діляться на п'ять каналів, які в сукупності охоплюють тестування управління інформацією та даними, рівні безпекової обізнаності персоналу, рівні управління шахрайством та соціальною інженерією, тестування комп'ютерних та телекомунікаційних мереж, бездротових пристроїв, мобільних телефонів, контроль безпеки фізичного доступу, контроль безпеки фізичних місць розташування, таких як житлові будівлі, адміністративні об'єкти та військові бази.

Цей проект підтримується Інститутом безпеки та відкритих методологій (ISECOM) та підлягає регулярній експертній та міждисциплінарній перевірці. Проект OSSTMM, як і всі проекти від ISECOM, вільний від комерційного та політичного впливу. Надається фінансування для всіх проектів ISECOM через партнерство, підписку, сертифікацію, ліцензування та дослідження на основі тематичних досліджень. ISECOM - це зареєстрована неприбуткова організація, створена в Нью-Йорку, США та в Каталонії, Іспанія.

OSSTMM зосереджується на технічних деталях процесу тестування на проникнення, містить детальні дослідження того, які саме елементи та об'єкти повинні бути перевірені в процесі тестування безпеки інформаційної системи, описує, що робити до, під час і після тестування безпеки та як виміряти результати проведеного тестування. OSSTMM також відомий своїми правилами ведення «бойових дій» (англ. Rules of Engagement), які визначають для тестувальника і клієнта, як тест повинен правильно проходити, починаючи із заперечення хибно-позитивних повідомлень від тестувальників про знайдені вразливості та закінчуючи тим, у якому вигляді замовнику слід очікувати отримання звіту [3]. Нові тести із кращої міжнародної практики кібербезпеки, законодавча база, правила та актуальні загрози регулярно оновлюються у кожній новій версії OSSTMM.

1.3.2 NIST

Організація NIST (National Institute of Standards and Technology) підіймає питання тестування на проникнення у документі SP800-115. Методологія NIST є менш вичерпною, ніж OSSTMM, проте вона є зрозумілішою для сприйняття регулюючими органами країн світу. З цієї причини NIST посилається на OSSTMM. Структура кібербезпеки Національного інституту стандартів і технологій відома в галузі кібербезпеки як золотий стандарт для керівництва комп’ютерною безпекою, вона може оцінювати та покращувати здатність організацій запобігати, виявляти та реагувати на кібератаки [4].

NIST Risk Management Framework — це структура, яка використовується для оцінки ризиків за параметрами NIST і може використовуватися для повідомлення про кіберризики керівникам бізнесу та персоналу, який працює з порушенням норм та правил інформаційної безпеки. Ці дві платформи працюють у тандемі, щоб створити всеосяжний протокол управління ризиками, який можна підлаштувати та пристосувати до потреб будь-якої компанії. З огляду на кількість інформації, що міститься у ньому, та його комплексність, NIST RMF (NIST SP 800-30) є одним із найскладніших у реалізації.

Метою публікації 800-30 є проведення оцінки ризиків безпеки відповідно до стандартів та рекомендацій NIST. NIST 800-30 спеціально використовується для пояснення кіберризиків у спосіб, зрозумілий людям, які не стикаються безпосередньо з галуззю кібербезпеки, наприклад, керівництву великої компанії. Спроможність пояснити складні речі в галузі кібербезпеки простими словами допомагає обом сторонам – і спеціалістам, і керуючому персоналу - приймати більш обґрунтовані рішення стосовно розподілу бюджету задля реалізації ініціатив із забезпечення кібербезпеки. Розподіл коштів на імплементацію бізнес-рішень з кібербезпеки виражається через типи загроз, з якими можна зіткнутися, можливий вплив реалізованих загроз безпеки на бізнес та фінансовий вплив на доходність компанії тощо. Для цього необхідна базова оцінка ризиків, щоб оцінити поточні стандарти роботи вже працюючих інформаційних систем та внести покращення, а також визначити, наскільки ці рішення впливають на цілісність даної ініціативи з покращення кібербезпеки.

Важливо мати рішення для підтримки усіх інформаційних систем компанії в належному стані безпеки, які включатимуть в себе процеси моніторингу та аналізуватимуть ті чи інші індикатори безпеки. Для цього необхідні апаратне забезпечення, програмне забезпечення, системні інтерфейси, дані про всі системи інформаційних технологій, інформація про те, наскільки витоки інформації є чутливими в конкретному випадку, інформація про тих, хто має доступ до інформаційних систем підприємства, а також інформація про цілі застосування та

функціональну складову інформаційних систем. Крім того, важливо мати в належному стані історію останніх загроз безпеки, а також звітність про попередні та поточні вразливості. Все це є необхідним для встановлення векторів загроз і враховуватиметься при створенні фінального звіту про тестування на проникнення. Далі, згідно NIST 800-30, буде розроблено список поточних і майбутніх запланованих заходів задля виправлення вже існуючих вразливостей безпеки та недопущення появи нових вразливостей. Ці процеси проводяться, щоб визначити можливі слабкі сторони безпеки інформаційних систем як відправну точку для покращення на основі позиціонування життєвого циклу розробки та функціонування IT-систем підприємства.

1.3.3 OWASP Testing Guide

На даний час найбільш популярною методологією проведення тестування на проникнення вебзастосунків є OWASP Testing Guide. OWASP використовує різні ресурси (людей, технології, процеси), щоб вирішити існуючі проблеми, що виникають у процесі розробки безпечних вебзастосунків. Це відбувається за допомогою впровадження практик безпеки та застосування інструментів безпеки, що надаються OWASP. Довгостроковий успіх проекту забезпечується тим, що всі учасники організації OWASP є добровольцями, включаючи саму раду директорів OWASP, керівників відділень, проектів та самих учасників цієї спільноти [5].

Після збору даних від більш ніж 40 відомих галузевих компаній із забезпечення безпеки застосунків, а також проведення опитування 500 видатних експертів у галузі кібербезпеки, OWASP опублікувала у мережі Top 10 Web Application Security Risks. 10 найнебезпечніших вразливостей було визначено на основі зібраної інформації та виявлених проблем у більш ніж 100000 різних застосунків.

OWASP Top 10 в основному фокусується на виявленні найсерйозніших вразливостей безпеки саме у вебзастосунках. Про кожну з вразливостей OWASP

надає загальну інформацію, що включає дані про основні принципи аналізу ймовірності виникнення вразливості та впливу на бізнес (BIA). Організація використовує наступний рейтинг, який ґрунтується на методології оцінки ризиків OWASP. OWASP Top 10 були відібрані відповідно до класифікації Common Weakness Enumeration (CWE) [6]. Декілька категорій змінилися порівняно з попереднім оглядом топ десятки вразливостей OWASP.

Згідно висновку OWASP Top 10 цього року, найбільш розповсюдженими вразливостями є:

– ін’єкції: можливість до ін'єкції на сайті (SQL, NoSQL, ОС, LDAP) з'являються, якщо недостовірні дані, які надсилаються інтерпретатору, є частиною деякої команди чи запиту. Такі дані, введені зловмисником, можуть змусити інтерпретатор користувацького вводу виконувати несанкціоновані команди та запити або допомогти отримати доступ до даних без належного дозволу;

– проблеми зі аутентифікацією: функції програми, пов’язані з аутентифікацією та керуванням сеансом, часто реалізуються неправильно, що дозволяє зловмисникам викрадати паролі, ключі та куки або використовувати інші недоліки в реалізації аутентифікації, щоб отримувати доступ до акаунтів адміністратора та інших користувачів тимчасово або на постійній основі;

– незахищеність критичних даних користувачів: багато вебзастосунків та API не захищають належним чином конфіденційні дані. Зловмисники можуть вкрасти або змінити слабко захищені дані, щоб вчинити фінансово-мотивоване шахрайство, крадіжку особи або інший злочин;

– зовнішні об’єкти XML (XXE): на старих вебресурсах, та на процесорах-XML з небезпечними конфігураціями, виконують оцінку URL-посилань на сторонні ресурси в XML документах. Ці посилання можуть потенційно можуть бути використаними зловмисниками, для отримання доступу до внутрішніх файлів за допомогою обробника URI, обміну внутрішніми файлами, сканування внутрішніх портів, віддаленого виконання коду та відмови в сервісних атаках;

– порушений контроль доступу: обмеження щодо дозволених користувачів, які мають певні повноваження змінювати в системи, часто не виконуються належним чином. Зловмисники можуть використовувати ці недоліки для доступу до несанкціонованих функціональних можливостей та/або даних, таких як доступ до облікових записів інших користувачів, перегляд конфіденційних файлів, зміна даних інших користувачів, зміна прав доступу тощо;

– неправильна конфігурація безпеки: ця вразливість безпеки є однією з найчастішою проблемою. Зазвичай це результат небезпечних конфігурацій за замовчуванням, неповних або спеціальних конфігурацій, відкритого хмарного сховища, неправильно налаштованих заголовків HTTP та багатослівних повідомлень про помилки, що містять конфіденційну інформацію. Не тільки всі операційні системи, рамки, бібліотеки та додатки повинні бути надійно налаштовані, але вони повинні бути виправлені/модернізовані своєчасно;

– міжсайтовий скриптинг (XSS): вразливість XSS виникає щоразу, коли програма включає недовірені дані на новій вебсторінці без належної перевірки чи видалення, або оновлює наявну вебсторінку із наданими користувачем даними за допомогою API браузера, який може створювати HTML або JavaScript. Ця вразливість, дозволяє злочинцю запустити свій скрипт в браузерах жертв. Зазвичай вони перехоплюють сеанси, видаляють контент вебсайта, або використовуються для переправлення трафіка на інші, часто шкідливі вебсайти;

– небезпепечна десереалізація: разливість призводить до віддаленого виконання коду. Навіть якщо недоліки десеріалізації не призводять до віддаленого виконання коду, їх можна використовувати для виконання інших атак, включаючи атаки відтворення стороннього контенту, атаки ін'єкції та атаки на зміну привілеїв доступу;

– використання компонентів, з відомими вразливими місцями: компоненти, такі як бібліотеки, рамки та інші програмні модулі, працюють із тими ж привілеями, що і програма. Якщо використовується вразливий компонент, така атака може полегшити серйозну втрату даних або захоплення сервера. Програми

та API, що використовують компоненти з відомою вразливістю, можуть підірвати захисні програми та включити різні атаки та впливи;

– недостатній облік даних та моніторинг: дана вразливість у поєднанні з відсутньою або неефективною інтеграцією з реакцією на інцидент дозволяє зловмисникам надалі атакувати системи, підтримувати стійкість, перемикатись на більшість систем, а також підробляти, витягувати або знищувати дані. Більшість досліджень щодо порушення моніторингу подій інформаційних систем свідчать про те, що час виявлення здійсненої кібератаки часто сягає понад 200 днів і, як правило, факти кібератаки виявляються зовнішніми сторонами, а не внутрішніми перевірками чи моніторингом.

OWASP Testing Guide є методологією перевірки на наявність вразливостей вебзастосунків, яка в тому числі може бути використана при тестуванні на проникнення вебзастосунків та систем, які використовують вебтехнології. Організацію OWASP (Open Web Application Security Project) було засновано у 2001 році, вона є спільнотою для розробників та спеціалістів кібербезпеки, яка працює над підвищенням безпеки програмного забезпечення за допомогою проектів програмного забезпечення з відкритим кодом. OWASP організовує провідні освітні та навчальні програми в галузі кібербезпеки, щоб тисячі учасників цих програм могли гарантувати, що залишаються «на плаву» серед поточних загроз безпеки.

Одним з продуктів є WSTG — це повний посібник з тестування безпеки вебзастосунків і вебсервісів. Створений спільними зусиллями професіоналів з кібербезпеки та відданих волонтерів, WSTG надає основу найкращих практик, які використовуються тестувальниками на проникнення та організаціями по всьому світу.

1.3.4 PTES

PTES (The Penetration Testing Execution Standard) є стандартом виконання тестування на проникнення, що складається з семи основних розділів. Вони охоплюють все, що пов’язано з тестуванням на проникнення – від початкової комунікації з клієнтами та аргументації необхідності проведення пентестів до збору інформації та моделювання загроз, у процесі чого пентестери намагаються краще зрозуміти «внутрішню кухню» організації, інформаційні системи якої перевіряються, до пошуку та дослідження вразливостей, експлуатації та пост-експлуатації, де досвід пентестерів поєднується з бізнес-розумінням поставленого перед ними завдання, і, нарешті, зі звітністю, яка містить детальний опис всіх процесів тестування на проникнення у спосіб, який має найбільший сенс для клієнта та найвище значення для нього, оскільки є беззаперечним доказом виконаної роботи тестувальників безпеки та містить всю необхідну інформацію по виявленим вразливостям, а також методам і рекомендаціям по їх усуненню [7].

Доступну версію PTES можна вважати версією 1.0, оскільки основні елементи стандарту пройшли «дорожні випробування» протягом більше року в галузі. Версія 2.0 ще розробляється, вона міститиме більш детальні відомості та рекомендації з точки зору "рівнів" - рівнів інтенсивності тестування, на яких кожен з елементів тесту на проникнення може бути виконаний. Оскільки жоден пентест не схожий на інший, а тестування безпеки буде варіюватися від тестування вебзастосунків до мережевої інфраструктури, зазначені рівні дозволятимуть організації-клієнту визначити, наскільки інтенсивні та обширні тестування безпеки слід проводити. Деякі початкові відомості про виконану роботу можна буде побачити в розділі збору даних, який є найпершим етапом у процесі проведення тестування безпеки.

Нижче наведено основні розділи, визначені стандартом PTES як основа для виконання тестування на проникнення:

– взаємодія перед початком тестування;

– збір розвідувальних даних;

– моделювання загроз;

– аналіз уразливостей;

– експлуатація;

– постексплуатація;

– звітність.

Оскільки стандарт не містить жодних технічних вказівок щодо виконання фактичного пентесту – було створено технічний посібник для супроводу самого стандарту.

1.3.5 MITRE ATT&CK

MITRE ATT&CK є глобально доступною базою знань про тактики та засоби атакуючого інформаційні системи, заснована на спостереженнях у реальному світі. База знань ATT&CK використовується як основа для розробки конкретних моделей і методологій загроз у приватному секторі, в державному секторі, а також у спільноті продуктів і послуг кібербезпеки. Створивши ATT&CK, організація MITRE виконує свою місію щодо вирішення проблем для безпечнішого світу інформаційних технологій, об’єднуючи спільноту для розвитку ефективнішої кібербезпеки. ATT&CK відкритий і доступний будь-якій особі чи організації для використання безкоштовно. Базу знань MITRE ATT&CK компанія MITRE створила у 2013 році. Мета проекту — складання структурованої матриці рішень, що використовуються кіберзлочинцями, щоб спростити завдання реагування на інциденти спеціалістами з кібербезпеки.

Інформація в базі знань MITRE ATT&CK представлена у вигляді матриць. Кожна матриця є таблицею, в якій заголовки стовпців відповідають тактикам кіберзлочинців, тобто основним етапам кібератаки або підготовки до неї, а вміст осередків - методикам реалізації цих тактик. Так, якщо збір даних згідно MITRE ATT&CK - це тактика атаки, то способи збору, наприклад, автоматичний збір або збір даних зі знімних носіїв - це техніки.

Матриці MITRE ATT&CK об'єднані у чотири групи:

– PRE-ATTCK: тактики та техніки, які зловмисники використовують на етапі підготовки до кібератаки;

– Enterprise: тактики та техніки, які зловмисники застосовують у ході атаки на підприємства. У цій групі доступна як зведена матриця, так і окремі матриці, що містять тактики та техніки кібератак на великі корпоративні системи та хмарні послуги;

– Mobile: тактики та техніки, які зловмисники використовують під час атаки на мобільні пристрої під керуванням iOS та Android;

– ATTCK for ICS: тактики та техніки, які використовуються в атаках на промислові системи управління.

Крім матриць, у базі знань MITRE ATT&CK доступні переліки технік, якими користуються відомі APT-угруповання, а також списки зловмисного інструментарію цих угруповань. Також, на сайті MITRE ATT&CK представлені основні методи зміцнення захисту інформаційних систем організацій та підприємств. Фахівці з інформаційної безпеки використовують матриці MITRE ATT&CK для вирішення наступних завдань:

– аналіз існуючого захисту на предмет відповідності реальним загрозам та підвищення безпеки інфраструктури компанії. За допомогою матриць MITRE ATT&CK можна визначити, до яких технік вразливі ресурси організації, щоб у короткостроковій перспективі усунути найкритичніші проблеми;

– своєчасне реагування на інциденти: за допомогою матриць MITRE ATT&CK можна встановити, на якому етапі розвитку знаходиться атака зловмисників і які контрзаходи необхідно вжити в першу чергу;

– розслідування кіберінцидентів: матриці MITRE ATT&CK дозволяють оперативно визначити, на якому етапі виявлено атаку і де варто в першу чергу шукати сліди вторгнення;

– атрибуція атак: за переліком технік, використаних зловмисниками, можна визначити імовірного виконавця;

– аналіз діяльності кіберзлочинців: матриці MITRE ATT&CK дозволяють відстежувати еволюцію тактик та технік, які застосовують відомі APT-угруповання;

– обмін інформацією із колегами: єдина структурована система опису кібератак дозволяє фахівцям із різних галузей знаходити спільну мову.

1.3.6 ISSAF

ISSAF є дуже хорошим довідковим джерелом інформації відносно тестування на проникнення, хоча станом на зараз ISSAF не є спільнотою, що активно розвивається [8]. Дане джерело надає комплексні технічні рекомендації щодо тестування на проникнення. Він охоплює такі теми як:

– управління проектами;

– рекомендації та найкращі практики безпеки;

– попередня оцінка захищеності інформаційної системи та післяоцінка по факту проведення тестування на проникнення;

– огляд політики інформаційної безпеки та організації безпеки об’єктів тестування;

– стратегії зламу інформаційних систем;

– оцінка безпеки систем на базі ОС Unix/Linux;

– оцінка безпеки систем на базі ОС Windows;

– оцінка безпеки баз даних;

– оцінка безпеки систем бездротового зв'язку.

За останній рік чи близько того, було зроблено значний поштовх до стандартизації термінології та таксономії. Остання версія ISSAF нормалізувала свої принципи з принципами основних галузевих постулатів, при цьому відкинувши один або два принципи, присутні в першому виданні Посібника з розробки. Це робиться для запобігання плутанини та підвищення дотримання основних стандартів.

Висновки до розділу 1

У розділі 1 було проведене дослідження такого явища, як тестування на проникнення: досліджувалися процеси, які виконуються при проведенні такого тестування, досліджувалися очікувані від таких тестувань результати, а також можливі проблеми у процесі проведення такого тестування та способи їх вирішення.

Було досліджено існуючі методології тестування на проникнення: історію їх виникнення, схожості та відмінності від інших методологій, цільове використання даних методологій в залежності від замовника тестування на проникнення та власне об'єкта тестування.

Було виконано порівняльний аналіз існуючих методологій для тестування на проникнення, які наразі можна застосовувати у процесі проведення пентесту вебзастосунку, у результаті чого зроблено висновок, що не всі методології є універсальними, актуальними та зручними у використанні й імплементації.

Найбільш універсальною, актуальною та зручною є методологія OWASP Testing Guide, адже вона постійно оновлюється експертами у галузі кібербезпеки, має наочні та зрозумілі пояснення та рекомендації та проста в імплементації саме в контексті створення та адміністрування безпечних застосунків.

2 АНАЛІЗ АКТУАЛЬНОСТІ ПРОБЛЕМИ КІБЕРАТАК В УКРАЇНІ. ДОСЛІДЖЕННЯ МОДЕЛІ TAKE-GRANT

2.1 Актуальність проблеми кібератак до повномасштабної війни

Відкритому збройному конфлікту між Росією та Україною передував період кібервійни, який розпочався у 2014 році.

Перші хакерські атаки на IT-системи державних установ та приватних компаній були зафіксовані під час Євромайдану. Після військової операції з анексії Криму Росія розпочала кібервійну проти України. Українські експерти також називають російські атаки кібервійною [9].

Загрози для України в кіберпросторі можна розділити на два основні рівні: перший вимагає лише новітніх IT-технологій і націлений на окремих осіб, а не на держави. На другому рівні домінують хактивізм, шпигунство та диверсії - злочини, характерні для геополітичної боротьби (або злочини регіонального рівня, які можуть впливати на політичну ситуацію в державі). Водночас, методи атак мають багато спільного.

Перший масштабний випадок хактивізму, з яким зіткнулася Україна, був пов'язаний із закриттям файлообмінника ex.ua. Після виведення з ладу файлообмінника правоохоронними органами були здійснені DDoS-атаки на урядові сайти, в тому числі на сайт Президента України та сайт Міністерства внутрішніх справ. Ці події вперше показали, що Україна не була ідеологічно та технологічно готова до таких атак. Реальних висновків з цих інцидентів зробити не вдалося, оскільки прямих економічних збитків не було, а те, що Україна не готова ефективно протистояти російській агресії в кіберпросторі, стало очевидним [10].

Після початку російсько-української війни у 2014 році компанії, що спеціалізуються на наданні послуг з кібербезпеки, почали фіксувати збільшення кількості кібератак на інформаційні системи нашої країни. У більшості випадків кібератаки спрямовані на викрадення життєво важливої інформації та можуть

надати противнику перевагу на полі бою. Російські кібератаки спрямовані на урядові установи України, ЄС та США, міністерства оборони, міжнародні та регіональні оборонні та політичні організації, аналітичні центри та засоби масової інформації [11].

З початком російсько-української війни в Україні, Росії та інших країнах з'явилися антиукраїнські хактивістські групи, які називають себе "Кіберберкут", "Анонімус" та інші. Не беручи до уваги складність визначення ступеня співпраці між хакерськими групами та органами державної влади, на основі зібраних доказів можна стверджувати, що проросійські хакерські групи здебільшого розташовані в Росії, а їхня діяльність спрямованана користь держави-окупанта та її державної політики.

Можна стверджувати, що з часом українська спільнота кібербезпеки розвинула навички виявлення, моніторингу та захисту від російських хакерських угруповань. Серед можливих пояснень - те, що з ескалацією конфлікту російські хакери мають менше часу для своєчасного оновлення та вдосконалення своєї тактики, техніки та злочинних методів, а також те, що Україна зараз тісніше співпрацює з міжнародними партнерами у сфері кібербезпеки.

Дослідники FireEye ідентифікували дві російські хакерські групи, які беруть активну участь у кібервійні між Росією та Україною: APT29 (також відоме як Cozy Bear і Cozy Duke) і APT28 (також відоме як Sofacy Group, Tsar Team, Pawn Storm і Fancy Bear) [12].

У 2013-2014 роках інформаційні системи деяких українських державних установ були заражені комп'ютерними вірусами, відомими як Snake/Uroborus/Turla. Таке шкідливе програмне забезпечення є дуже складним, стійким до контрзаходів і, ймовірно, було створене у 2005 році [13].

У 2013 році противник розпочав операцію "Армагеддон" - російську кібершпигунську кампанію проти інформаційних систем українських державних органів, правоохоронних та оборонних структур; 23 грудня 2015 року було підтверджено першу глобальну атаку, спрямовану на виведення з ладу

енергосистеми. Російські хакери успішно атакували комп'ютерну систему управління в диспетчерській "Прикарпаттяобленерго", вимкнувши близько 30 підстанцій і залишивши близько 230000 жителів без електрики на термін від однієї до шести годин. Атака була здійснена за допомогою шкідливого програмного забезпечення BlackEnergy. Одночасно були атаковані "Чернівціобленерго" та "Київобленерго", але вони зазнали меншої шкоди.

Наступні кібератаки відбулися в ніч з суботи 17 грудня 2016 року на неділю 18 грудня 2016 року. Збій в автоматиці управління на підстанції "Північна" залишив без електроенергії споживачів північного правого берега Києва та сусідніх областей. Основною версією інциденту стала кібератака - зовнішнє втручання через мережу передачі даних. Оператор підстанцій "Укренерго" спочатку не підтвердив факт кібератаки, але експерти, залучені до розслідування, підтвердили, що відключення було спричинене кібератакою. Проміжні результати розслідування були представлені експертами Information Systems Security Partners Олексієм Ясинським та Мариною Кротофіль з Honeywell Industrial Cyber Security Lab на конференції S4 у Флориді, США, 10 січня 2017 року. Втім, зловмисники не завдали значної шкоди, натомість атака мала на меті "демонстрацію сили". Як і в попередніх випадках, ця атака була частиною великої фішингової операції проти українських урядових організацій: 6 грудня 2016 року хакерські атаки на урядові вебсайти (наприклад, Національного казначейства України) та внутрішні мережі державних установ призвели до значних затримок у бюджетних платежах. Вже 7 грудня Кабінет Міністрів виділив 80 мільйонів гривень на боротьбу з хакерами.

27 червня 2017 року було розпочато масштабну хакерську атаку на компанії різних типів і розмірів власності, державні та неурядові організації з використанням комп'ютерного хробака Petya. Цей новий зразок шкідливого програмного забезпечення дослідники називають по-різному: DiskSoder.C, ExPetr, PetrWrap, Petya або NotPetya [14].

У жовтні 2017 року відбулася масована атака вірусу-черв'яка BadRabbit. Спочатку жертви були інфіковані через низку скомпрометованих вебсайтів.

Дослідники вважають, що за цим могли стояти розробники вірусу NotPetya. Незабаром після цього голова української кіберполіції Сергій Демедюк заявив, що атака з використанням вірусу Bad Rabbit була прикриттям для більш складної атаки на компанії, що використовують російське програмне забезпечення 1С.

14 січня 2022 року було атаковано близько 22 державних організацій та 70 українських вебсайтів. На постраждалих сайтах були розміщені тексти польською, українською та російською мовами, які засуджували український націоналізм. Зображення вказували на Польщу як на джерело атаки, але польський текст містив значні граматичні помилки. Вважається, що це була російська хакерська атака, спрямована на залякування і створення загрози для Польщі [15].

Аналогічна DDoS-атака на українські вебсайти була здійснена 15 лютого 2022 року о 20:21, яку, на думку української влади, здійснила Росія. Близько 15 банківських сайтів та доменів gov.ua були виведені з ладу на п'ять годин; атаковані були ПриватБанк та Ощадбанк, а також сайти Міністерства оборони, Збройних сил та Міністерства з питань реінтеграції тимчасово окупованих територій. За словами міністра Михайла Федорова, 15 лютого відбулася найбільша кібератака в історії України, яка завдала мільйонних збитків. Однак деякі експерти стверджують, що вартість атаки не перевищила кількох тисяч доларів.

23 лютого 2022 року, за день до російського вторгнення в Україну, відбулася чергова атака на урядові та банківські сайти: близько 16:00 були зламані сайти Верховної Ради, Кабінету Міністрів України, Міністерства закордонних справ та Служби безпеки України. Міністерство освіти і науки закрило доступ до своїх сайтів, щоб запобігти кібератакам. Офіційні особи США пов'язали атаку з Росією; за даними ESET, атака стала можливою завдяки зараженню сотень комп'ютерів вірусом HermeticWiper, який був зібраний 28 грудня 2021 року.

Аналізуючи останні кілька років, можна виявити певні закономірності та проаналізувати тенденції щодо кількості та ефективності ворожих кібероперацій в Україні. Наприклад, за даними Служби безпеки України, у квітні 2021 року фахівці Служби виявили 1,5 мільйона підозрілих інцидентів інформаційної безпеки та 68,5 тисячі потенційних кіберінцидентів [16]. Вони також припинили 53 значні кіберінциденти. У травні 2021 року кількість підозрілих інцидентів зросла до 3,1 мільйона, а кількість потенційних кіберінцидентів зменшилася до 65,6 тисячі. СБУ також нейтралізувала 24 критичні інциденти: 1,8 млн у червні, 2,2 млн у липні, 1,2 млн у серпні, 2,4 млн у вересні, 2,5 млн у жовтні, 2,1 млн у листопаді, 2,3 млн у грудні, у червні виявлено 76 підозрілих інформаційних подій, у липні - 65, у серпні - 43, у вересні - 53, у жовтні - 53, у листопаді – 57, у листопаді - 53, у грудні - 59, а у січні 2022 року - 6,8 млн підозрілих подій та 25,5 тис. потенційних кіберподій. Крім того, фахівці попередили 121 великий кіберінцидент.

На рис. 2.1 наведено повну інфографіку інцидентів безпеки за період з квітня 2021 року по січень 2022 року.

Рисунок 2.1 – Інфографіка інцидентів безпеки від СБУ станом на 15.02.2022

2.2 Актуальність проблеми кібератак після початку повномасштабної війни

В ночі та вранці 24 лютого 2022 року, з початком тотальної окупації України, сайти Київської обласної державної адміністрації зазнали хакерської атаки, а деякі ресурси були відключені для захисту даних. Державна служба спеціального зв'язку та захисту інформації виявила велику кількість електронних листів на сайтах i.ua та meta.ua, які містили фішингові посилання на приватні адреси українських військових та чиновників. Зловмисники використовували протокол IMAP для компрометації поштових серверів та отримання електронних адрес для подальшої розсилки. За даними ДССЗЗІ, ці атаки були здійснені білорусами з мінського угруповання UNC1151, до складу якого входять офіцери Міністерства оборони Республіки Білорусь.

За новою інформацією, актуальною на 31 січня 2023 року, у період з вересня та по кінець грудня 2022 року, пов'язані з Росією APT-угруповання продовжували брати активну участь в операціях, спрямованих проти України, застосовуючи руйнівні програми-вайпери та програми-вимагачі. Протягом цього періоду, в Україні було виявлено сумнозвісне угруповання Sandworm, яке використовувало раніше невідоме шкідливе програмне забезпечення проти компаній енергетичного сектору. Зазвичай, керівництва країн або фінансовані державою суб'єкти керують APT-групами, фінансуючи їх та спрямовуючи їх зусилля; описана атака сталася в жовтні, в той самий період, коли російські збройні сили почали наносити ракетні удари по об'єктах енергетичної інфраструктури України. Хоча дослідники компанії ESET, які опублікували дану інформацію, не можуть довести факт того, що ці події були скоординовані, вони припускають, що Sandworm і російські військові мають спільні цілі [17].

ESET назвала новітню програму-вайпер із серії раніше виявлених схожих об’єктів ПЗ - NikoWiper. Цей вайпер був використаний проти одного з підприємств енергетичного сектору в Україні в жовтні 2022 року. NikoWiper

базується на SDelete, утиліті командного рядка від Microsoft, яка використовується для безпечного видалення файлів.

Окрім шкідливих програм для знищення даних - вайперів, ESET виявила факти здійснення атак за допомогою ПЗ Sandworm, які використовували програми-вимагачі в якості вайперів. Хоча в цих атаках використовувалися програми-вимагачі, їхня кінцева мета була такою ж, як і у вайперів: знищення даних. На відміну від традиційних атак з використанням програм-вимагачів, у рамках яких зловмисники спочатку заражають жертв своїм ПЗ, шифруючи всі файли користувача, а потім за значну суму надають код дешифрування цих файлів, оператори Sandworm не мають наміру надавати ключ розшифровки [18].

У жовтні 2022 року ESET виявила розгортання програми-вимагача Prestige проти логістичних компаній в Україні та Польщі, а в листопаді 2022 року ESET виявила в Україні нове ПЗ-вимагач, написане на .NET, яке отримало назву RansomBoggs. Поряд із Sandworm, інші російські групи APT, такі як Callisto та Gamaredon, продовжували свої кампанії шпигунства проти України з метою викрадення облікових даних користувачів та встановлення бекдорів.

2.3 Дослідження моделі Take-Grant в розрізі даної бакалаврської кваліфікаційної роботи

У багатьох моделях дискреційного доступу виявилась проблема несанкціонованого поширення прав доступу. У 1976 році була запропонована модель Take–Grant. Основними елементами цієї моделі є графи доступів і правила їх перетворень. Призначенням моделі Take–Grant є аналіз шляхів розповсюдження прав доступу по вихідному графу прав доступу у системах дискреційного розмежування доступу. Модель допускає наявність прав доступу не лише у суб’єктів до об’єктів, але і в об’єктів до об’єктів. У подальшому з’явилась розширена модель Take–Grant, яка стала одним із методів дослідження захищених систем.

Детально розглядаючи елементи моделі Take–Grant, їх можна визначити наступним чином:

– O – множина об’єктів системи;

– $\subseteq$ O – множина суб’єктів системи;

– E – множина встановлених прав доступу суб’єкта x до об’єкта y із правом α, де $\alpha \in$ R = {r1,r2,...,r_m}$\cup${t, g} – множина видів прав доступу, t (take) – повноваження брати права доступу, g (grant) – повноваження надавати права доступу.

Основу моделі Take–Grant складає скінченний орієнтований граф без петель G$\subseteq$ (S,O,E), який визначає поточні доступи в системі. У цьому графі елементи множин S і O є вершинами графу, які позначаються як: $\otimes$ – об’єкти (елементи множини O\S), □ – суб’єкти (елементи множини S). Елементи множини E $\subseteq$ O × O × R є ребрами графу, кожне з яких помічено непустою підмножиною множини видів прав доступу R, $\alpha i \in$ R.

Порядок переходу із стану до стану системи визначається операціями або правилами перетворення графу доступів. Перетворення графа G в граф G' у результаті виконання правила op позначимо через G ├op G'. У класичній моделі Take–Grant розглядаються чотири де-юре правила перетворення графа.

Виконання кожного з правил може бути ініційовано лише суб’єктом, який є активною компонентою системи:

– правило take(α,x,y,z) – право брати права доступу;

– правило grant(α,x,y,z) – право надавати права доступу;

– правило create(β,x,y) – право створювати новий об’єкт;

– правило remove(α,x,y) – право видалити права доступу на об’єкт.

Правило take(α,x,y,z)

Нехай x $\in$ S, y, z $\in$ O – різні вершини графу G, $\beta \subseteq$ R, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графа доступів G′ з графу G.

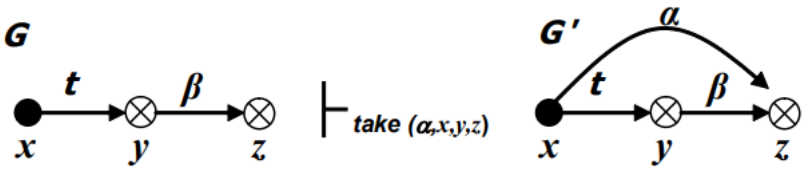

Рисунок 2.1 – Суб'єкт x бере в об'єкта y права $\alpha \subseteq \beta$ на об'єкт z

Правило grant(α,x,y,z)

Нехай $x \in S$, $y, z \in O$ – різні вершини графу G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графу доступів G' із графу G.

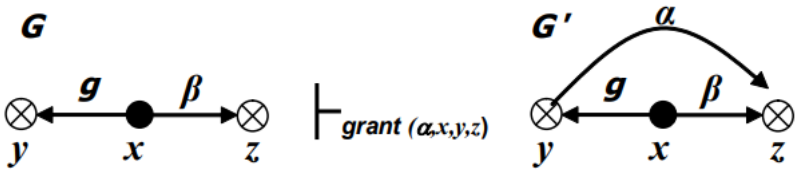

Рисунок 2.2 – Суб'єкт x надає об'єкту y права $\alpha \subseteq \beta$ на об'єкт z

Правило create(β,x,y)

Нехай $x \in S$, $\beta \subseteq R$, $\beta \neq \varnothing$. Правило визначає порядок одержання нового графу G´ з графу G; $y \in O$ – новий об’єкт або суб’єкт.

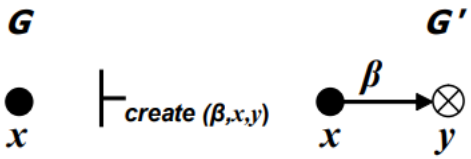

Рисунок 2.3 – Суб'єкт x створює новий β – доступний об'єкт y

Правило remove(α,x,y)

Нехай $x \in S$, $y \in O$ – різні вершини графу G, $\beta \subseteq R$, $\alpha \subseteq \beta$. Правило визначає порядок одержання нового графа G′ з графа G.

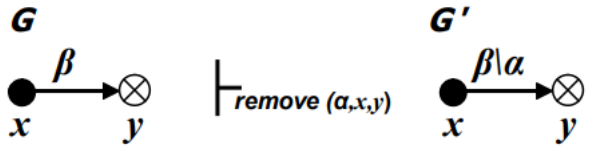

Рисунок 2.4 – Суб'єкт x видаляє право доступу α на об'єкт y

Методом представлення даних для даної моделі є граф доступу. Метою моделювання являється аналіз шляхів поширення прав доступу.

Розглянемо модель Take-Grant на прикладі тестового середовища від Portswigger. Логінимось у акаунт користувача під ніком wiener та паролем peter:

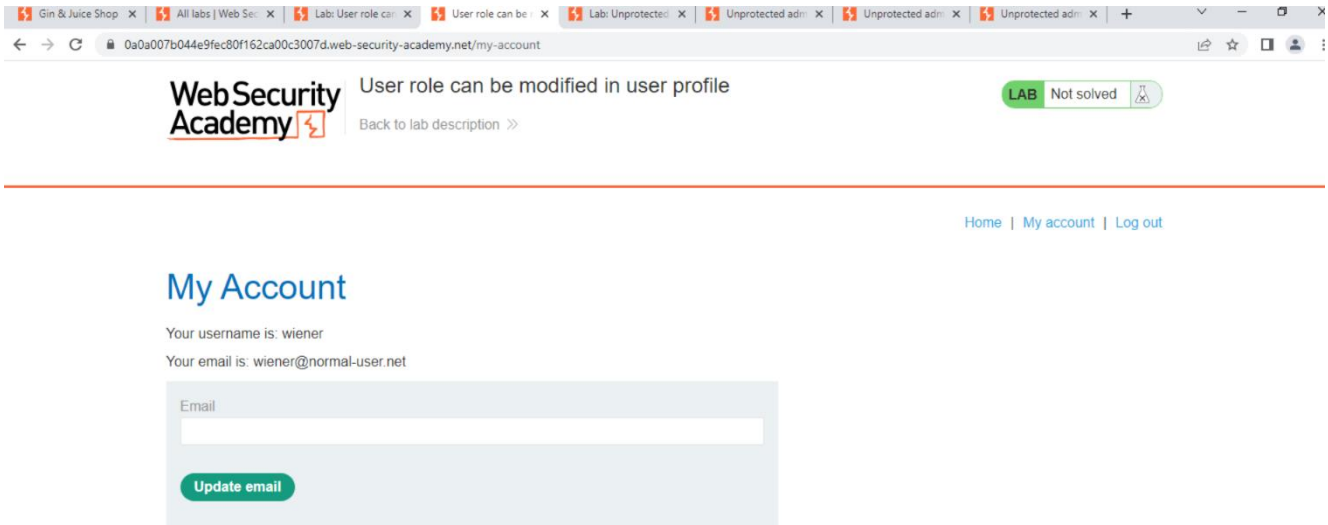

Рисунок 2.5 – Тестове середовище для моделі Take-Grant

Завданням цього тестового застосунку є отримання доступу до адміністратора за допомогою модифікації значення параметру roleid у тілі HTTP-запиту. Змінюємо значення roleid та отримуємо акаунт адміністратора:

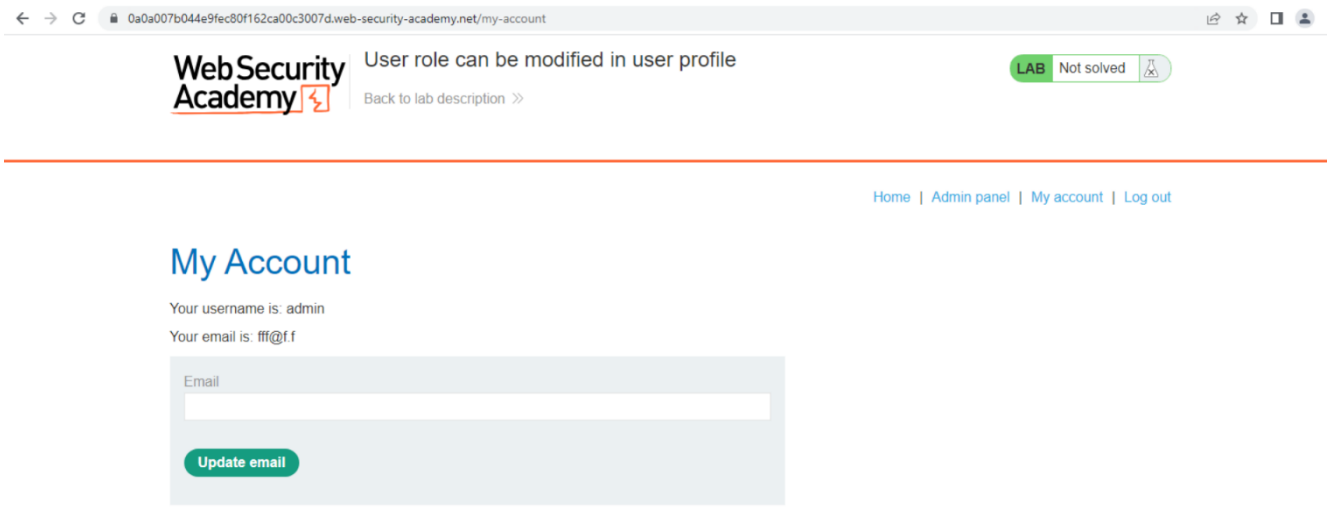

Рисунок 2.6 – Отримання ролі адміністратора

Опираючись на функціонал застосунку та існуючі категорії доступу до об’єктів, виводимо наступне:

O (множина об’єктів системи) = 9, а саме головна сторінка, список товарів, кнопка повернення до опису тестового середовища, кнопка переходу до налаштувань акаунта, повернення на домашню сторінку, вихід з акаунта, функціонал зміни поштової скриньки, опис товарів, функціонал видалення користувача (доступне адміністратору), $S \subseteq O$ (множина суб’єктів системи) = 2, оскільки в нормальних умовах маємо доступ до 1 звичайного користувача та 1 адміністратора з урахуванням, що за 1 акаунтом перебуває 1 персона. Так як суб'єктів – 2, то множин встановлених прав доступу суб'єкта x до об'єкта y із правом α у нас теж буде дві.

Позначимо суб’єкт «користувач» як x1, суб’єкт «адміністратор» як x2, множина встановлених прав доступу суб’єкта x1 до об’єкта y із правом α як E1, а множина встановлених прав доступу суб’єкта x2 до об’єкта y із правом α як E2. Підмножиною $\{r1,r2,...,r_m\}$ описуються доступні об'єкти для обох типів доступу – користувача та адміністратора. Множина видів прав доступу для звичайного користувача виглядає як $\alpha \in R = \{r1,r2,...,r_8\} \cup \{t\}$, а для адміністратора виглядає як $\alpha \in R = \{r1,r2,...,r_9\} \cup \{t\}$.

Повноваження надавати права доступу відсутні для обох категорій доступу в даному випадку, проте часто зустрічається реалізація адміністраторських привілеїв, в рамках яких можна надавати права доступу до тих чи інших об'єктів чи дій для звичайних користувачів.

Висновки до розділу 2

У ході проведення аналізу актуальності захисту вебзастосунків з огляду на збільшення кількості кібератак та інцидентів безпеки на Україну до початку повномасштабної війни та після її початку було встановлено, що починаючи з 2014 року, темпи та масштаби кібератак радикально зростали, особливо протягом останніх 2-3 років. Для дослідження використано статистичні дані з квітня 2021 року по січень 2022 року, дані про епізоди кібератак у лютому 2022, а також звітності про кібероперації ворога у період з вересня по кінець грудня 2022 року як спроби зовнішнього впливу Росії на енергетичний, адміністративний, державний, приватний сектори України з метою принесення максимально можливої економічної шкоди, викрадення цінної інформації про військово-політичне керівництво України та українських громадян в цілому, створення соціальної напруги в суспільстві та напруги між військово-політичним керівництвом України та її союзниками.

Росія в ході кібератак на Україну використовує широкий спектр програмних засобів, зокрема, при атаках на вебзастосунки. Найчастіше різноплановість інструментів зустрічається саме при проведенні ворожих кібероперацій хакерськими угрупованнями, що працюють на державні органи та мають у своєму членстві, більшою мірою, держслужбовців та кадрових військовослужбовців.

З початком повномасштабного вторгнення Україна отримала посилення кіберзахисту у вигляді як і надання необхідного програмного забезпечення країнами-партнерами, так і надання фахівців та експертів у галузі кібербезпеки, що дозволило ефективніше протистояти ворожим кіберопераціям, зменшити вже нанесену шкоду та заздалегідь планувати заходи безпеки задля швидкого

виявлення та реагування на інциденти безпеки або унеможливлення проведення ефективних кібератак на державний та приватний сектори України. Особлива увага з боку України також приділяється контрзаходам: щодня проводяться кібероперації відносно об’єктів російського держсектору з метою отримання корисної інформації щодо подальших оперативних та стратегічних планів ворога відносно України та інших держав. Також величезна робота, як публічна, так і негласна, відбувається відносно воєнно-промислового комплексу Росії з метою нанесення максимального ураження ВПК ворога задля унеможливлення проведення бойових дій на території України та швидшого закінчення війни.

Варто зробити висновок, що хоча кібератаки ворога направлені не тільки на ураження та злам вебзастосунків, частка атак на такий тип ресурсів величезна серед усього обсягу кібератак, тому потребує забезпечення максимально можливого захисту з нашого боку задля унеможливлення здійснення різнопланової шкоди як хакерськими угрупованнями, що працюють на державні та військові апарати ворогів України, так і на хакерів, що мають фінансову або особисту мотивацію задля здійснення своїх атак.

В результаті проведення дослідження математичної моделі для аналізу дискреційного розмежування доступу Take-Grant було виконано поставлені завдання в даному розділі, а саме проаналізовано, за якими правилами працює дана модель, й за цими правилами досліджено тестове середовище у вигляді вебзастосунку.

Під час виконання завдання було повністю описано усі правила дискреційного розмежування доступу на прикладі вебзастосунків, створених компанією Portswigger. Було застосовано знання з управління контролем доступу, надані освітньою платформою даної компанії [19].

Описані моделі дискреційного розмежування доступу мають практичне застосування не тільки у тестових середовищах з освітньою метою, але й і у проектуванні, створенні, управлінні та адмініструванні вебзастосунків різного рівня масштабу, складності та сфери застосування.

3 АНАЛІЗ СКАНЕРІВ ВРАЗЛИВОСТЕЙ БЕЗПЕКИ

3.1 Сканери вразливостей безпеки вебзастосунків як інструментарій для проведення тестування на проникнення

Спеціалісти в галузі кібербезпеки, а саме в проведенні тестування вебзастосунків на проникнення часто використовують автоматичні сканери вразливостей безпеки. Такі сканери використовують заздалегідь запрограмовані патерни пошуку вразливостей, адже нерідко трапляється так, що вразливості часто виникають за одних і тих самих причин, відповідно, і шукати їх можна за певними очікуваними патернами.

Важливо розуміти, що ніколи не можна сподіватися тільки на роботу сканерів вразливостей, оскільки вебзастосунки бувають дуже різними за своєю структурою, призначенням та функціоналом, що нерідко «збиває з пантелику» автоматизовані програмні засоби для аудиту безпеки, такі як сканери вразливостей. З цього якраз випливає необхідність людського втручання у процесс тестування вебзастосунків на проникнення, на наявність вразливостей безпеки [20].

Іншим важливим аспектом є факт того, що існують вразливості, які сучасні сканери вразливостей безпеки просто не в змозі ідентифікувати, тому, що ідентифікувати такі вразливості можна лише повністю розуміючи та усвідомлюючи, як працює вебзастосунок, який планується протестувати. До таких вразливостей відносять, наприклад, вразливості бізнес-логіки, знайти які складно часом навіть кваліфікованим експертам та фахівцям у галузі кібербезпеки.

Ще одним прикладом, який підтверджує думку про те, що сканер вразливостей безпеки – не панацея, є існування таких вразливостей, як request smuggling, response splitting та client-side desync [21]. Дані категорії вразливостей є новими у світі кібербезпеки, про що свідчить їх відносно нещодавня поява у профільних джерелах в Інтернеті та, як наслідок, невелика кількість інформації стосовно їх виявлення, експлуатації та усунення.

3.2 Порівняльний аналіз сканерів вразливостей безпеки вебзастосунків

3.2.1 Burp Suite

Після встановлення програмного продукту, відкриваємо його, обираємо варіант Temporary project та тиснемо Next. Далі обираємо варіант Use Burp defaults та тиснемо Start Burp.

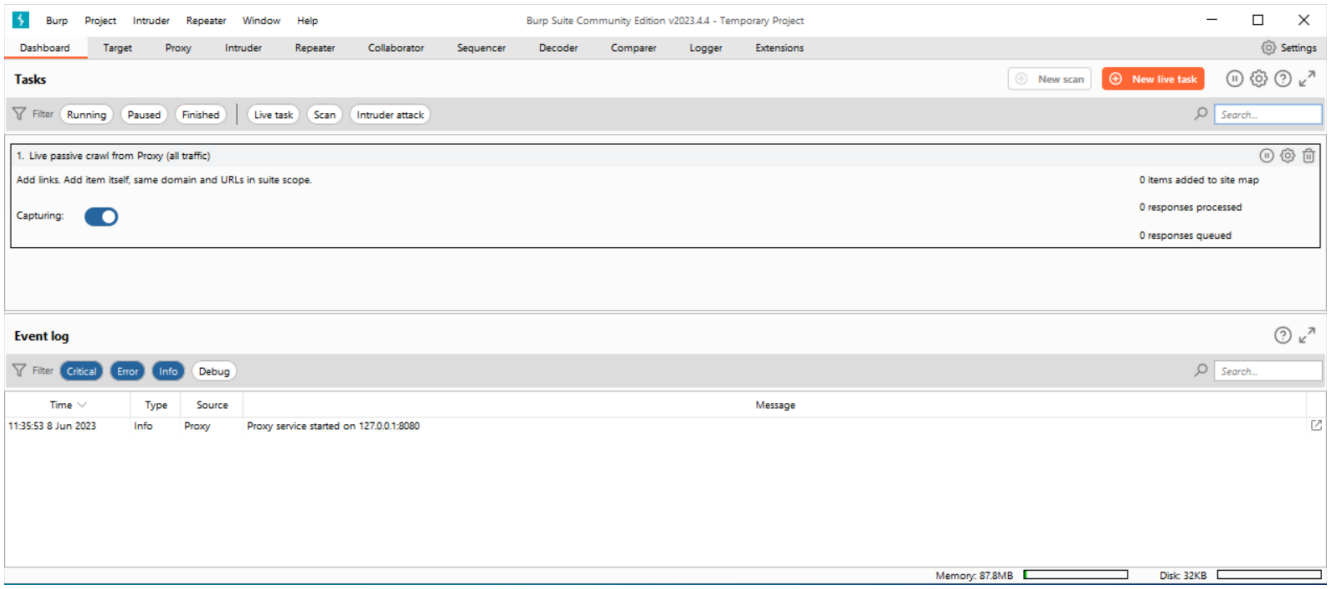

Рисунок 3.1 – Головне вікно ПЗ Burp Suite

Відкриваємо вкладку Proxy, обираємо підвкладку HTTP history. Знаходимо потрібний нам HTTP-запит:

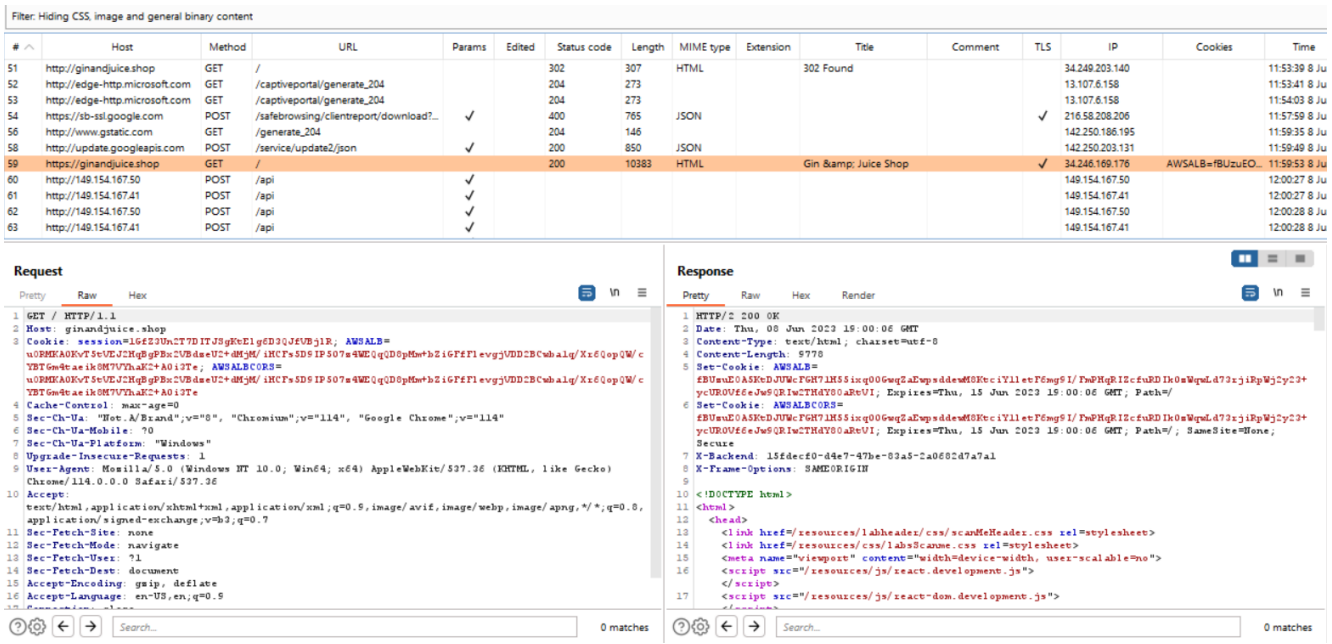

Рисунок 3.2 – Обрання HTTP-запиту для роботи з ним

Далі тицяємо правою кнопкою миші та обираємо Send to Repeater. У вкладці Repeater натискаємо кнопку Send та очікуємо на відповідь сервера. Бачимо код 200 OK – це означає, що головна сторінка вебзастосунку завантажилася без усіляких проблем:

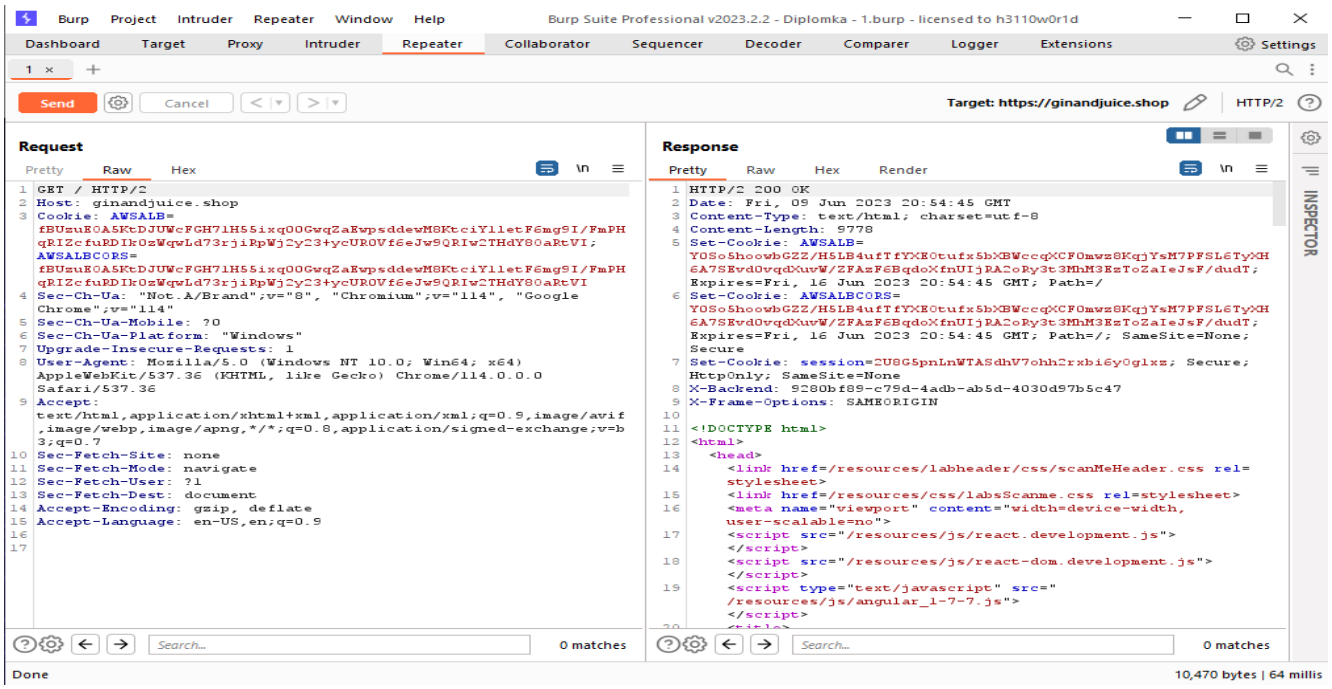

Рисунок 3.3 – Робота в інструменті Repeater

Тицяємо правою кнопкою миші по запиту, обираємо пункт Scan. Відкривається наступне вікно:

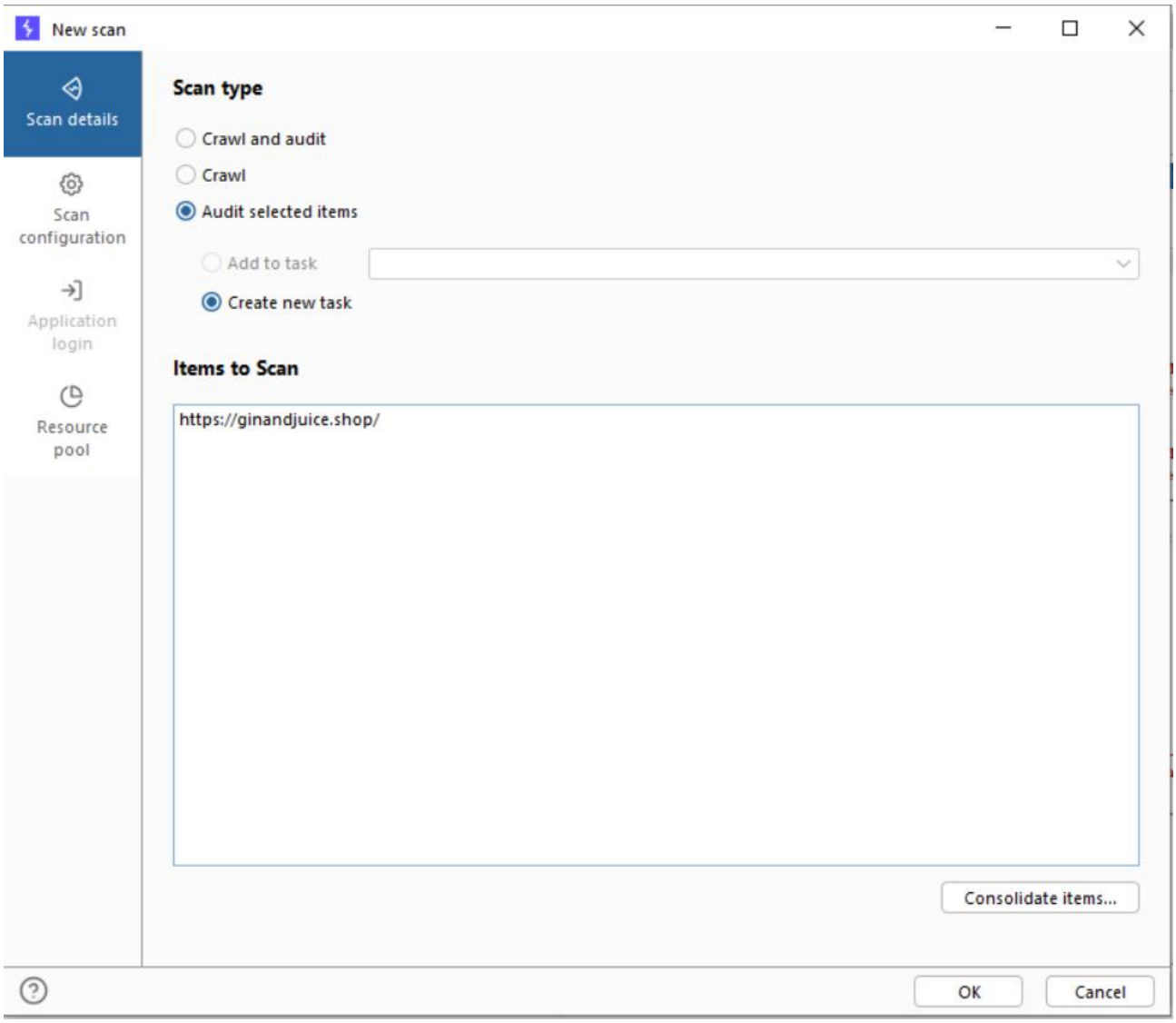

Рисунок 3.4 – Головне вікно налаштувань сканування

Обираємо опцію Crawl and Audit в Scan options. Більше нічого не змінюємо на даній вкладці, переміщуємося до вкладки Scan configuration. Обираємо варіант Use a custom configuration та налаштовуємо скан самостійно. Тицяємо на кнопку New… і далі Crawling:

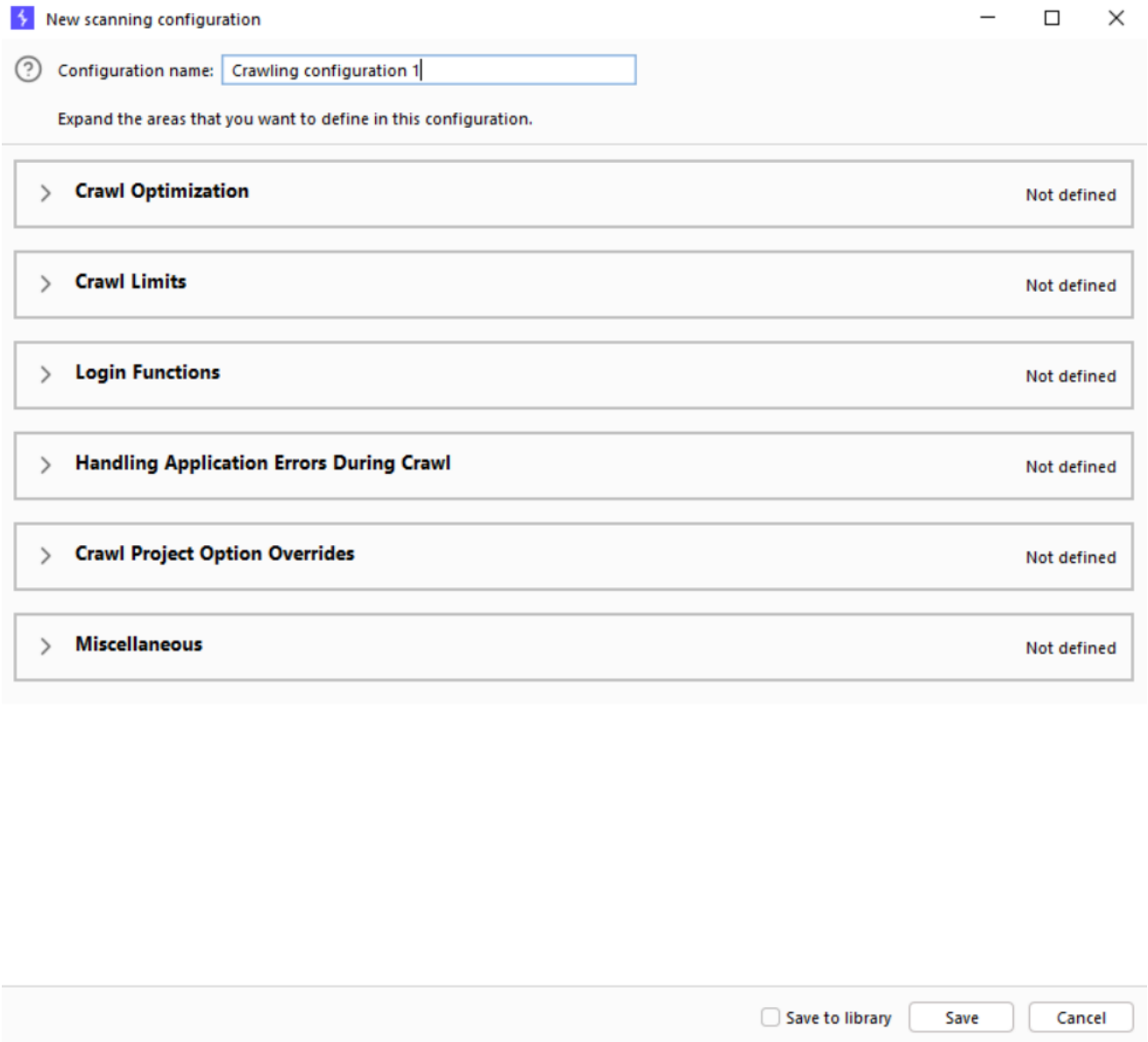

Рисунок 3.5 – Конфігурування crawling-функціоналу

Подальші налаштування виглядатимуть таким чином:

- блок Crawl Optimization: параметр Maximum link depth – значення 4, параметр Crawl strategy – значення Normal;
- блок Crawl Limits: параметр Maximum crawl time – значення 150, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;
- блок Login Functions: лишається незмінним;
- блок Handling Application Errors During Crawl: лишається незмінним;

– блок Crawl Project Option Overrides: лишається незмінним;
– блок Miscellaneous: лишається незмінним.

Зберігаємо налаштування кравлера.

Переходимо до налаштувань аудиту:

– блок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;
– блок Issues Reported: лишається незмінним;
– блок Handling Application Errors During Audit: лишається незмінним;
– блок Insertion Point Types: вибрати всі опції;
– блок Modifying Parameter Locations: вибрати всі опції;
– блок Ignored Insertion Points: лишається незмінним;
– блок Frequently Occurring Insertion Points: лишається незмінним;
– блок Misc Insertion Point Options: лишається незмінним;
– блок JavaScript Analysis: лишається незмінним;
– блок Audit Project Options Override: лишається незмінним;

Зберігаємо всі налаштування. Тиснемо кнопку ОК. Скан почався.

3.2.2 Acunetix Web Vulnerability Scanner

Після встановлення програмного продукту, відкриваємо браузер та вводимо у адресний рядок: https://localhost:3443. Відкривається сторінка логіну у акаунт адміністратора:

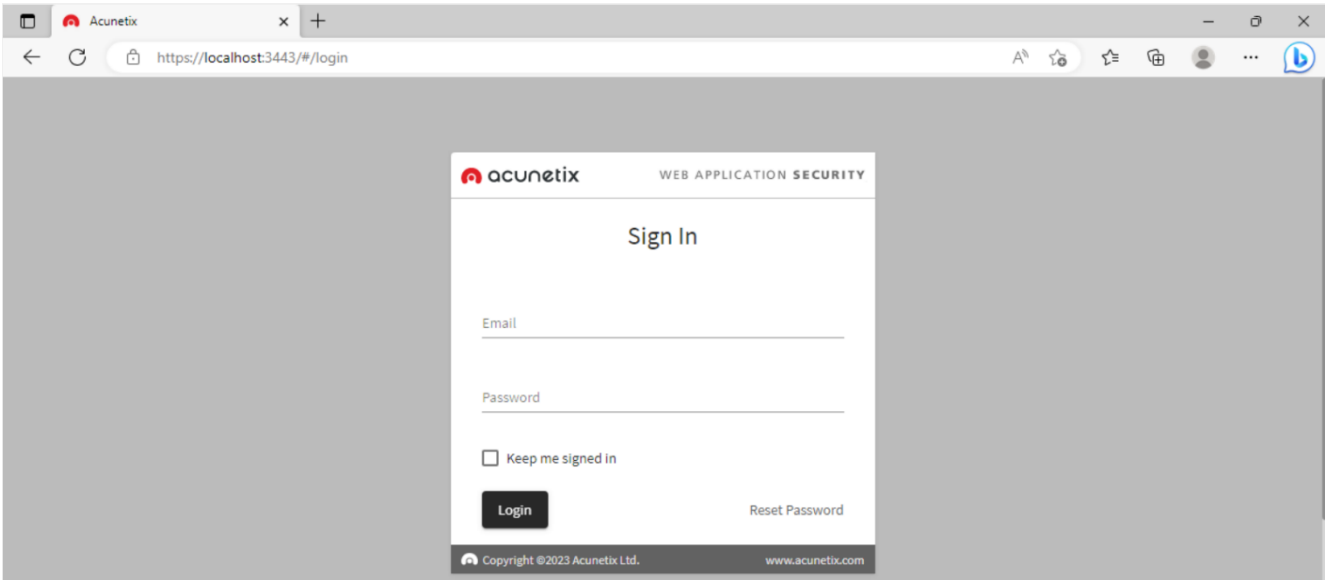

Рисунок 3.6 – Вікно логіну у застосунок

Після введення логіну та паролю адміністратора застосунок перекидає на головну сторінку, яка виглядає наступним чином:

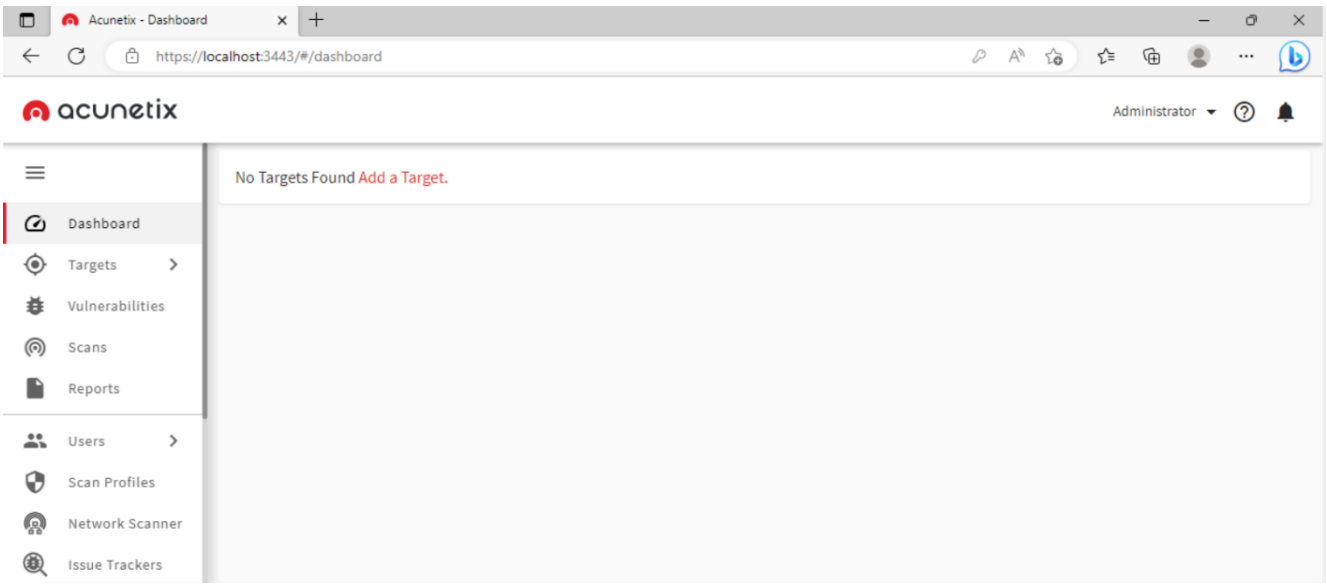

Рисунок 3.7 – Головне вікно застосунку

Дашборд включає в себе такі об’єкти, як: власне перехід на сторінку дашборду, список цілей для тестування, знайдені вразливості, управління сканами, менеджмент звітів по проведених скануваннях, управління

користувачами сканера, управління профілями сканів (в залежності від специфіки вебзастосунку має сенс застосовувати різні налаштування сканів, які зберігаються у профілях), налаштування сканування мереж (в рамках даної бакалаврської кваліфікаційної роботи цей пункт є неактуальним для використання, хоча демонструє обширність функціоналу сканера), налаштування синхронізації із трекерами багів, налаштування email-розсилок, управління сесіями, керування «годинами відпочинку» сканера, керування проксі-серверами та керування загальними налаштуваннями.

Клікнувши кнопку Add Target, на екрані з’являється наступне:

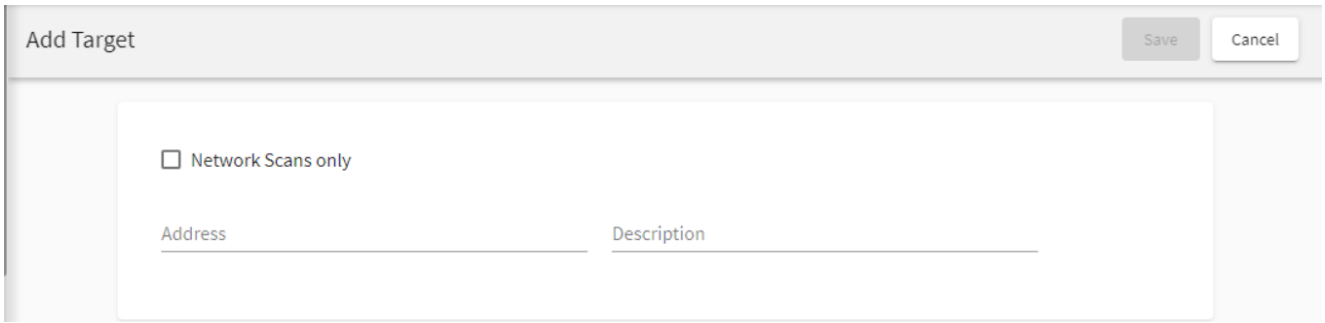

Рисунок 3.8 – Етап додавання «цілі» для тестування

У поле Address вводимо URL: https://ginandjuice.shop/ - це тестове середовище для сканерів безпеки вебзастосунків. Поле Description можна не чіпати, воно є опціональним і там можна лишати коментарі для орієнтування по сканах, якщо їх багато. Бокс Network Scans only теж не треба чіпати.

Натискаємо кнопку Save та бачимо наступне вікно:

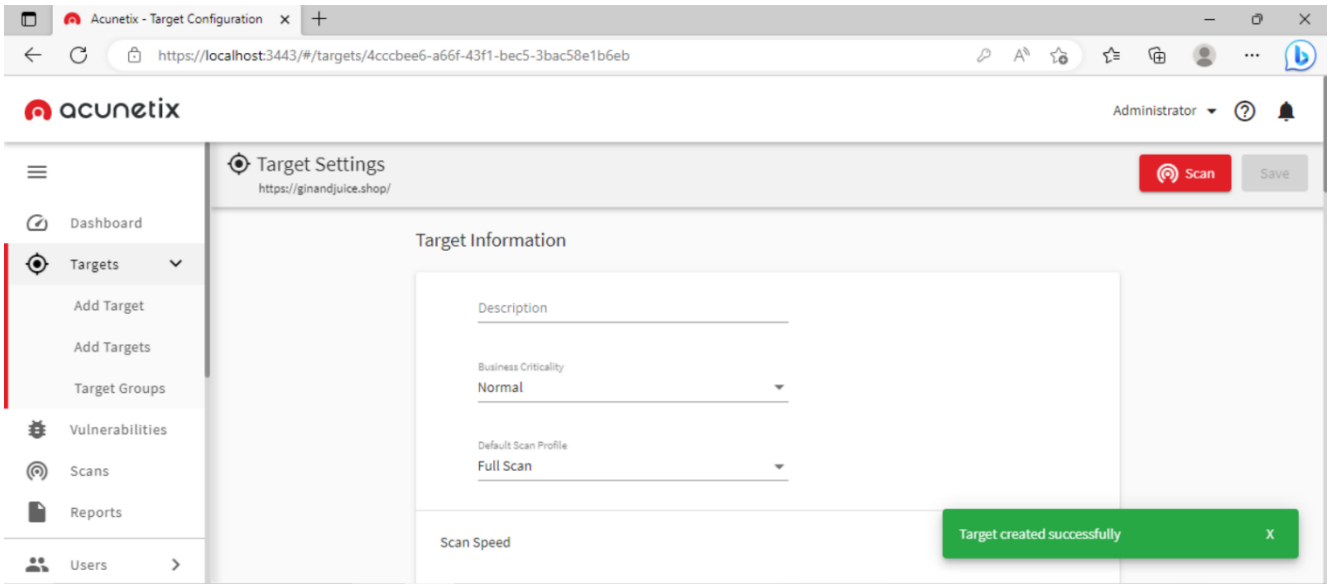

Рисунок 3.9 – Налаштування сканування

Поле Description є опціональним. Поле Business Criticality впливає на можливість фільтрувати сканування і вразливості на основі цієї метрики, що дозволить зосередитися на більш важливих об'єктах у вебзастосунку, якщо це необхідно – залишаємо поле як є. Поле Default Scan Profile впливає на пошук певних категорій вразливостей, а так як в даному прикладі ми проводимо аудит безпеки вперше та не знаємо, які вразливості можуть бути в застосунку – варто лишити дане поле незмінним. Поле Scan Speed впливає на кількість одночасно відправленних запитів на сайт за одиницю часу. Параметр Continuous Scanning впливає на можливість запуску сканування по певним дням автоматично. Параметр Site Login використовується для вказання данних для аутентифікації тестового користувача у вебзастосунку. Параметр Business Logic Recording використовується для запису макросів на випадок, якщо структура застосунку є дуже складною і треба «пояснити» сканеру, що у яких саме частинах застосунку треба шукати вразливості безпеки. Модуль AcuSensor дозволяє сканеру збирати більше інформації з вебзастосунків на базі PHP, .NET, Java або Node.js, що поліпшує результати сканування і зменшує кількість хибно-позитивних результатів. Параметр User-Agent дозволяє кастомізувати юзер-агент на випадок,

коли застосунок працює тільки у певному браузері тощо. Параметр Case Sensitive Paths впливає на відправлення запитів на сервер з урахуванням того, чи чутливі URL-шляхи застосунку до регістру, чи вони не є чутливими. Параметр Limit Crawling to address and sub-directories only впливає на те, чи варто сканеру обмежитися кроулінгом заданого URL вебзастосунку та піддиректорій, чи ні. Параметр Excluded Paths містить піддиректорії застосунку, які треба виключити з процесу тестування. Параметр HTTP Authentication впливає на можливість аутентифікації користувача за допомогою різних типів аутентифікації. Параметр Client Certificate дозволяє підвантажити власний SSL-сертифікат для застосунку у випадку, якщо не використовується якийсь загальновідомий. Параметр Proxy Server дозволяє налаштувати з'єднання з проксі-сервером, через який відбуватиметься процес тестування застосунку. Параметр Technologies дозволяє власноруч обрати мову програмування, якою написаний вебзастосунок. Параметри Custom Headers та Custom Cookies дозволяють вказати власноруч бажані HTTP-заголовки та куки відповідно. Параметр Issue Tracker дозволяє вказувати систему контролю версій задля подальшого моніторингу знайдених вразливостей [22].

Натискаємо кнопку Save, щоб зберегти налаштування, а потім тиснемо Scan. У віконці, що відкрилося, змінюємо тільки поле Report, де обираємо варіант Affected Items.

Тиснемо кнопку Create Scan. Далі побачимо наступне віконце:

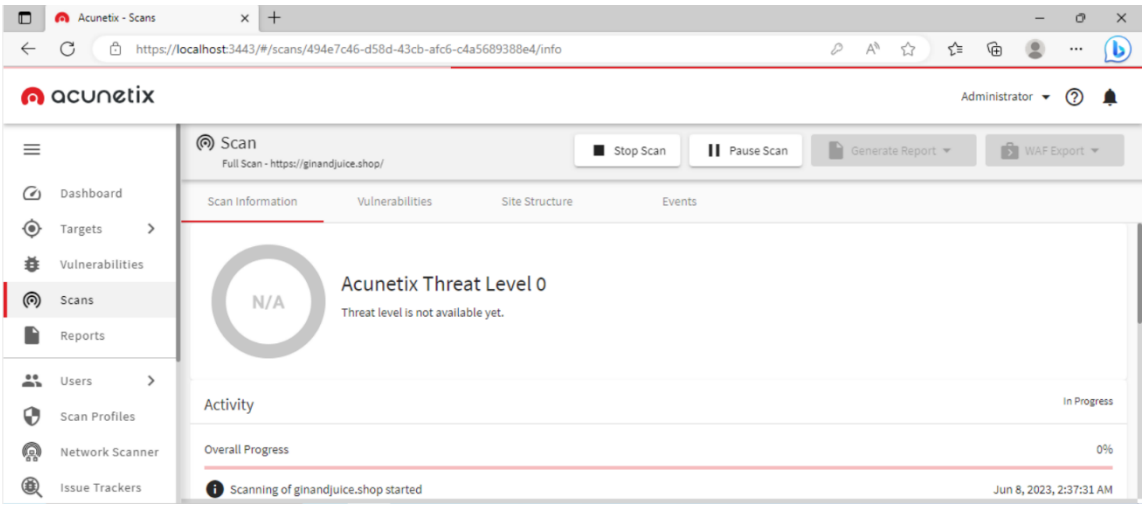

Рисунок 3.10 – Сканування розпочато

Через деякий час починаємо досліджувати результати тестування на проникнення вебзастосунку:

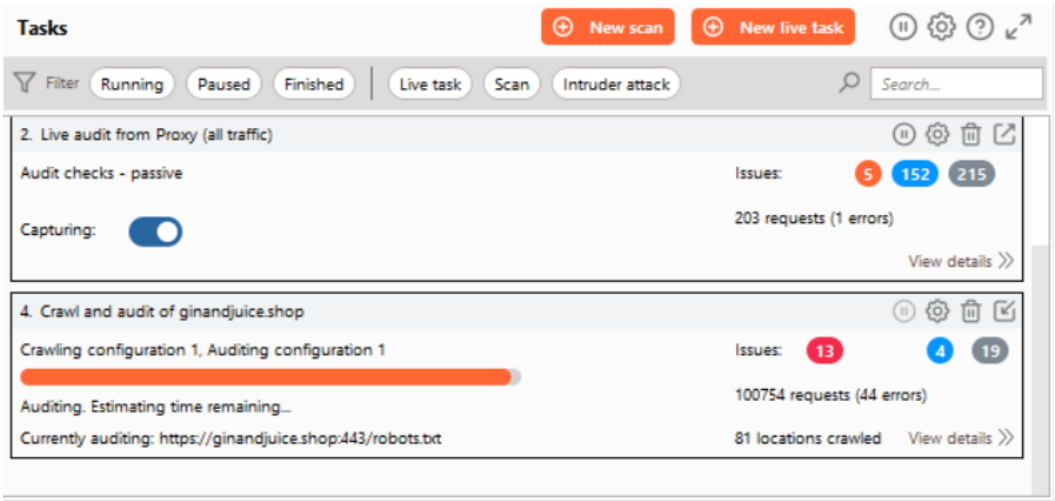

Рисунок 3.11 – Результати тестування у Burp Suite

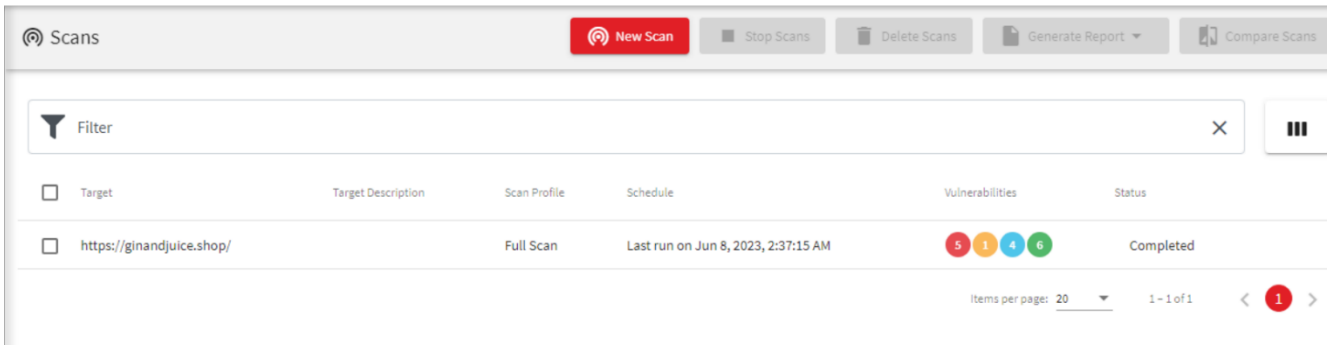

Рисунок 3.12 – Результати тестування в Acunetix

Детальні результати тестування на проникнення у Burp Suite виглядають наступним чином:

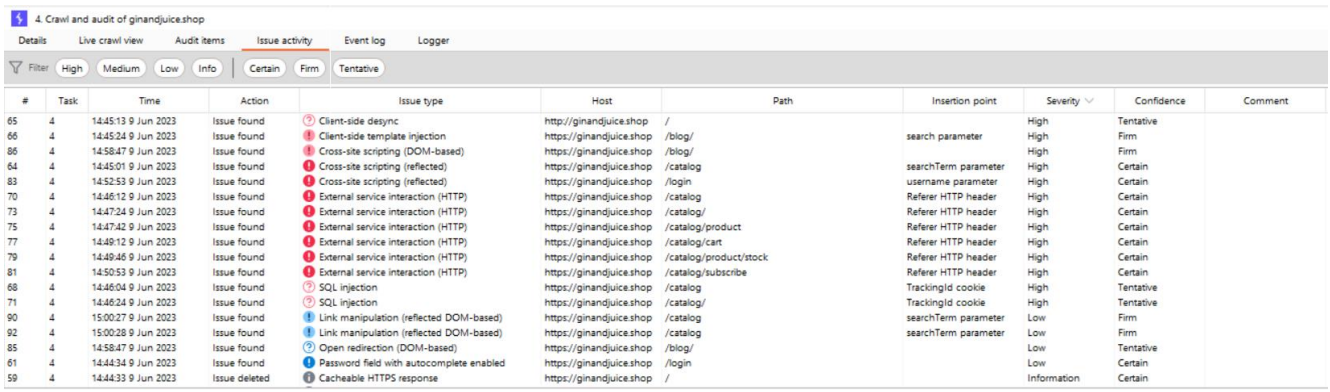

Рисунок 3.13 – Детальні результати

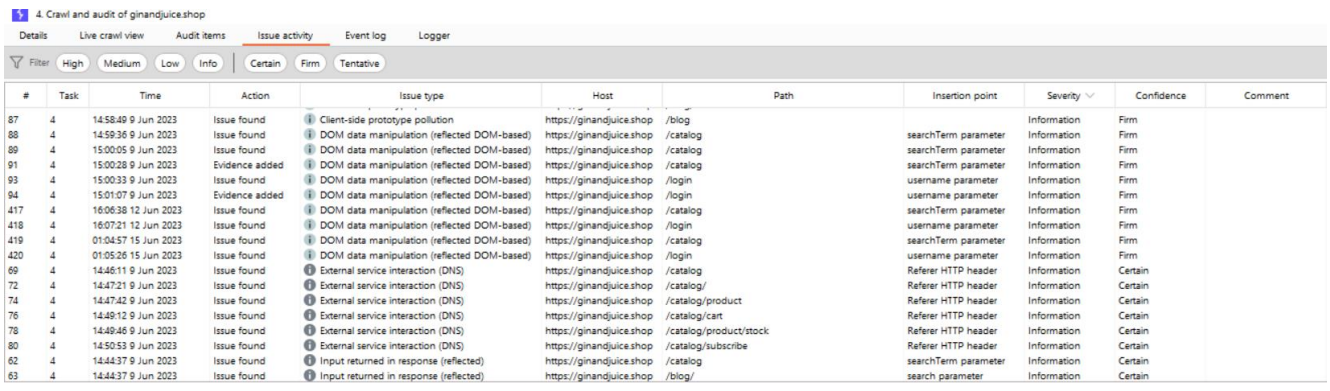

Рисунок 3.14 – Детальні результати

Детальні результати тестування на проникнення у Acunetix виглядають ось так:

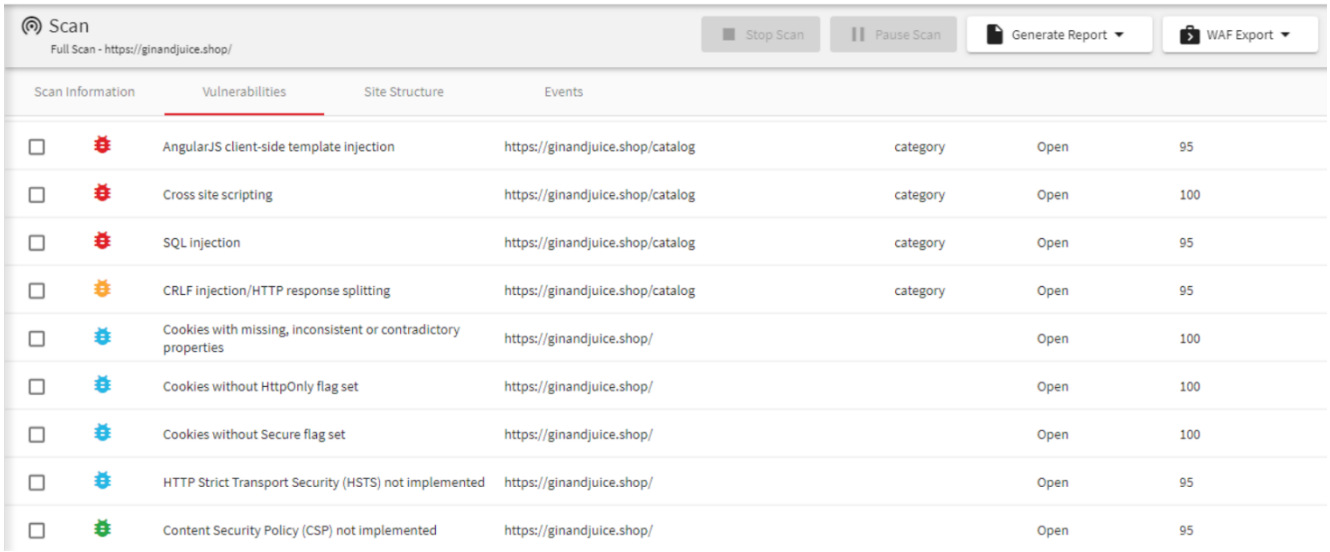

Рисунок 3.15 – Детальні результати

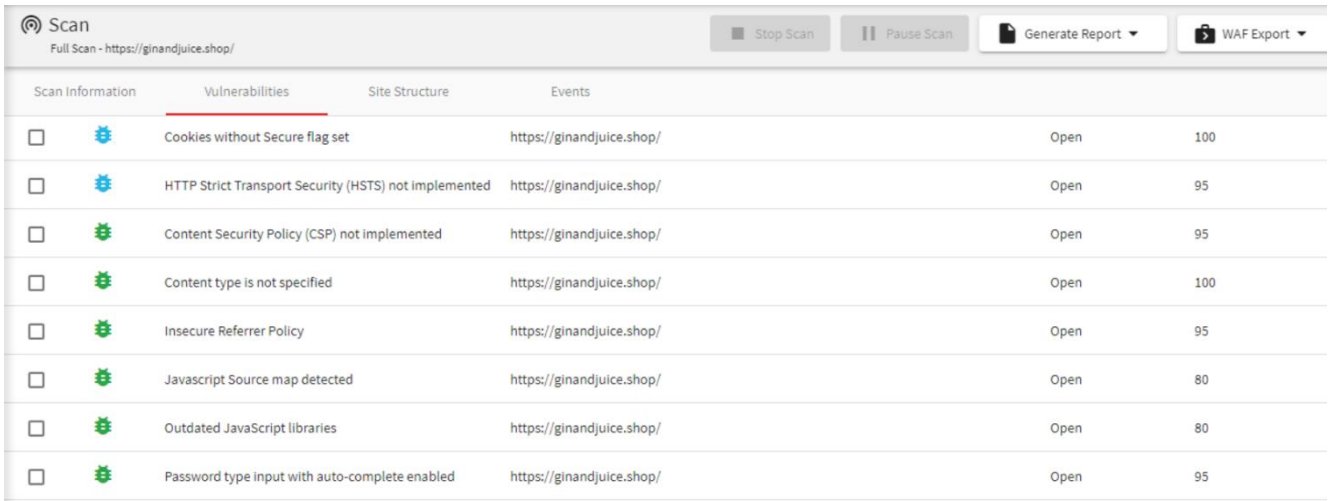

Рисунок 3.16 – Детальні результати

Всі знайдені вразливості категоризуємо по рівнях Critical, High, Medium, Low та Info:

Таблиця 3.1 – Всі знайдені вразливості безпеки

Програмне забезпечення для тестування на проникнення	Кількість знайдених вразливостей рівня Critical	Кількість знайдених вразливостей рівня High	Кількість знайдених вразливостей рівня Medium	Кількість знайдених вразливостей рівня Low	Кількість знайдених вразливостей рівня Info

Burp Suite (Community/Professional)	0	12	0	4	23
Acunetix Web Vulnerability Scanner	0	5	1	4	6

Не дивлячись на те, що кількість знайдених вразливостей безпеки більша у Burp Suite, аніж у Acunetix Web Vulnerability Scanner, варто розуміти, що нерідко у процесі сканування будь-який елемент на сторінці застосунку, будь-яка поведінка застосунку, або нестандартна реакція на HTTP-запити може «заплутати» сканер під час тестування, в результаті чого останній промаркує вразливість як валідну, але насправді вона може бути хибно-позитивною [23]. Тому дуже важливим аспектом є дослідження вищевказаних програмних продуктів саме з точки зору ефективності їх роботи, а це передбачає мануальне відсіювання хибно позитивних результатів аудиту безпеки від дійсних, валідних результатів аудиту безпеки.

3.3 Детальний аналіз ідентифікованих вразливостей безпеки. Порівняння роботи сканерів

3.3.1 Вразливість Client-side template injection

Першою будемо аналізувати вразливість Client-side template injection. Дана вразливість була ідентифікована обома програмними засобами. Характеризується дана вразливість тим, що у шаблон на стороні клієнта, який використовується додатком, можна вставляти довільні AngularJS-вирази. У параметрі search було передано корисне навантаження 804sg{{a=(7*7.0)}}e4c09. Це корисне навантаження було відображено без змін у HTTP-відповіді програми. Ехо-відгук з'являється в клієнтському шаблоні AngularJS, як зазначено в директиві "ng-app" у відповідному HTML-тезі. У фрагменті коду також видно, що використовується AngularJS версії 1.7.7. Ця атака демонструє, що можна вставляти довільні

AngularJS-вирази у поле пошуку та вони будуть відображатися на сторінці застосунку, що свідчить про те, що не відбувається жодної фільтрації користувацького введення на бекенді [24]. Зловмисник може використати це для AngularJS v1.7.7, щоб виконати довільний JavaScript-код в браузері цільового користувача. Не дивлячись на те, що програмним засобом Acunetix було ідентифіковано 3 екземпляри даної вразливості у ендпойнтах /blog, /login та /catalog, а програмним засобом Burp Suite було знайдено лише один екземпляр даної вразливості у ендпойнті /blog – при детальному аналізі виявилося, що валідною є лише екземпляр вразливості, знайдений ПЗ Burp Suite, інші ж екземпляри є хибно-позитивними.

3.3.2 Вразливість Reflected XSS

Наступною вразливістю для дослідження є Reflected XSS (Cross-site scripting). Ця вразливість також була ідентифікована обома програмними продуктами. Значення параметра HTTP-запиту searchTerm копіюється в рядок JavaScript, який береться в одинарні лапки. У параметрі searchTerm було передано корисне навантаження 92316\';alert(1)//241. У відповіді застосунку цей ввід було відображено так:

Рисунок 3.17 – Демонстрація вразливості у коді

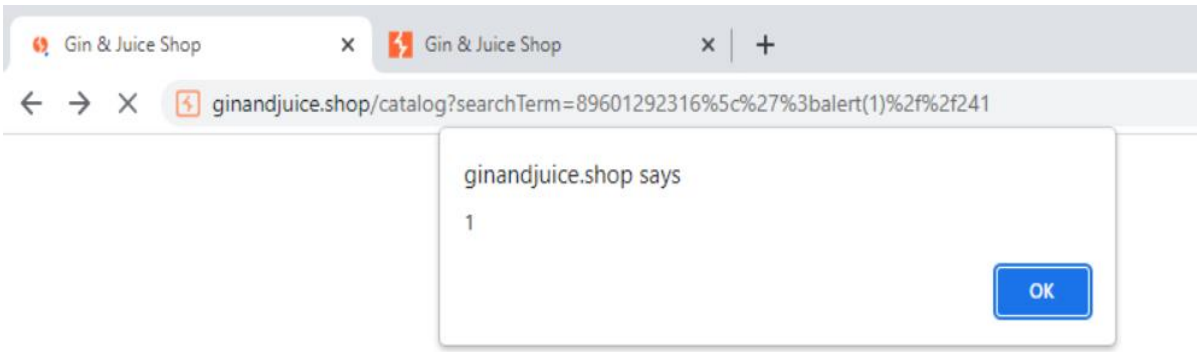

Рисунок 3.18 – Демонстрація вразливості у браузері

Ця атака демонструє, що у поля користувацького вводу застосунку можна вставити довільний JavaScript-код та він буде безперешкодно виконуватися [25]. Зазвичай, застосунок намагається запобігти завершенню цитованого рядка JavaScript, ставлячи символ зворотної косої риски (\) перед будь-якими символами лапок, що містяться у вхідних даних. Метою цього захисту є уникнення лапок і запобігання завершенню рядка. Однак програма не може екранувати символи зворотної косої риски, які вже містяться у вхідних даних. Це дозволяє зловмиснику поставити свій власний символ зворотної косої риски перед лапкою, який екранує символ зворотної косої риски, доданий програмою, і таким чином лапки залишаються не екранованими, що дозволяє виходити за межі вводу тексту та виконувати довільний JS-код [26]. Ця техніка використовується у продемонстрованій атаці.

Така сама вразливість була знайдена у ендпойнті /login, що було ідентифіковано сканером Burp Suite, але проігноровано сканером Acunetix:

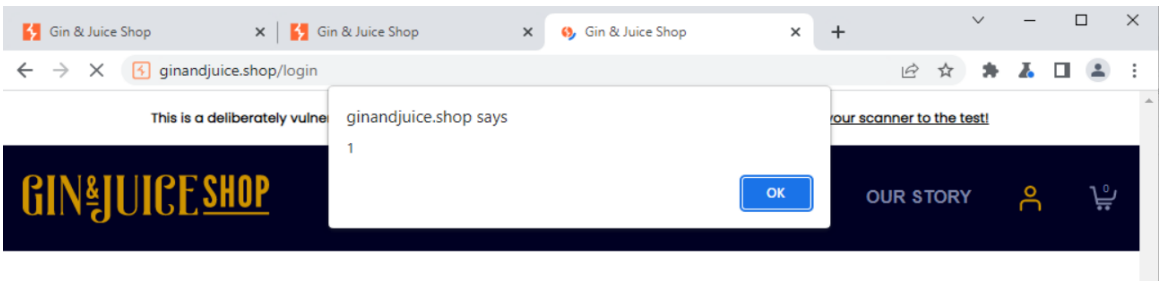

Рисунок 3.19 – Демонстрація вразливості у браузері

3.3.3 Вразливість DOM-based XSS

Далі розглядимемо ідентифіковану ПЗ Burp Suite вразливість DOM-based XSS. Вразливості на основі DOM виникають, коли скрипт на стороні клієнта зчитує дані з контрольованої частини DOM (наприклад, URL-адреси) і обробляє ці дані небезпечним чином. Міжсайтовий скриптинг на основі DOM виникає, коли скрипт записує контрольовані дані в HTML-документ у небезпечний спосіб. Зловмисник може використати цю вразливість для створення URL, яка, якщо її відвідає інший користувач програми, призведе до виконання коду JavaScript, наданого зловмисником, у браузері користувача в контексті сеансу роботи цього користувача з програмою [27]. Код, наданий зловмисником, може виконувати широкий спектр дій, таких як крадіжка сесії або облікових даних жертви, виконання довільних дій від імені жертви, реєстрація натискань клавіш тощо. Користувачів можна змусити перейти на сфабриковану зловмисником URL-адресу різними способами, подібно до звичайних векторів реалізації вразливостей міжсайтового скриптингу.

В описі даної вразливості у Burp Suite міститься proof-of-concept, який було скопійовано та вставлено у адресний рядок браузера:

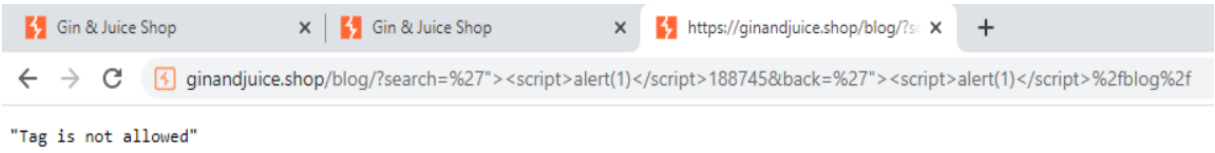

Рисунок 3.20 – Вразливість не є валідною

Отже, дослідивши цей кейс, є розуміння, що дана знахідка є хибно-позитивною і не дарма була проігнорована ПЗ Acunetix Web Vulnerability Scanner.

3.3.4 Вразливість Client-side desync

Розглянемо вразливість Client-side desync. Вразливості десинхронізації на стороні клієнта виникають, коли вебсервер не може коректно обробити заголовок Content-Length POST-запитів, що характеризує довжину HTTP-запиту в байтах із розрахунку 1 символ = 1 байт. Використовуючи цю поведінку, зловмисник може змусити браузер жертви десинхронізувати з'єднання з сайтом, що зазвичай призводить до XSS. Зазвичай, атака відбувається у кілька послідовних запитів до сервера: перший запит відправляється таким чином, що у першій його частині міститься POST-запит, а у другій – GET-запит [28]. У POST-запиті заголовок Content-Length повинен бути більшим за дійсну кількість символів у тілі запиту і якщо вебсервер пропускає такий запит – це означає, що вразливість можна спробувати реалізувати та поєднати з іншими вразливостями.

У даному ж випадку, перший запит виглядає таким чином:

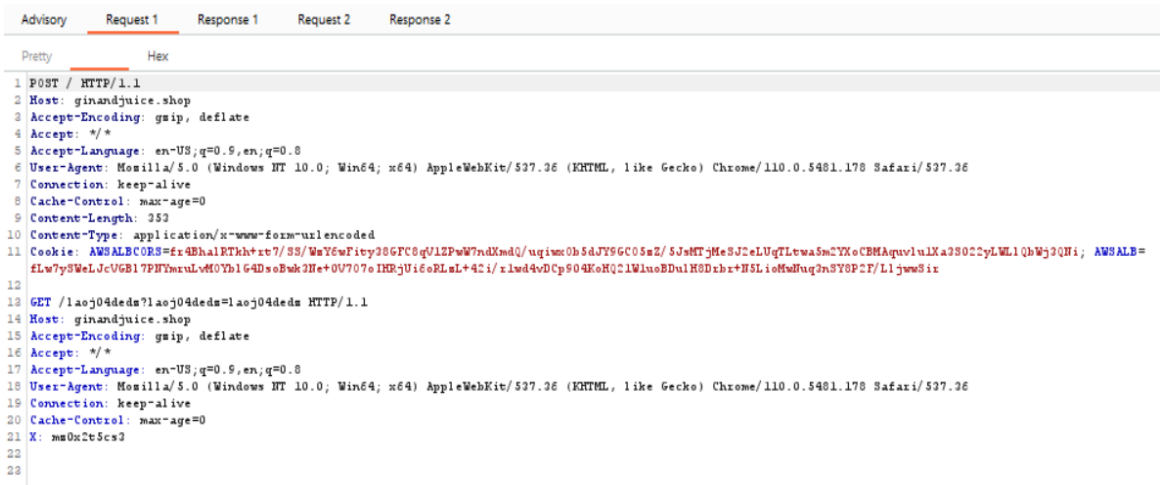

Рисунок 3.21 – Потенційно вразливий запит

Однак, відповідь застосунку на такий запит повертає статус-код 302:

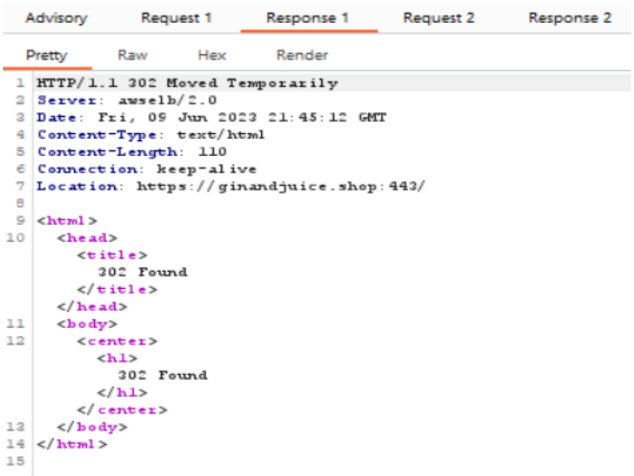

Рисунок 3.22 – Відповідь сервера

Це свідчить про те, що вразливість не є валідною. Очікуваним статус-кодом відповіді є код 200 ОК, однак в даному випадку маємо 302 Moved Temporarily. Все це свідчить про те, що ПЗ Burp Suite визначив цю знахідку вразливою, однак насправді вона такою не є. ПЗ Acunetix Web Vulnerability Scanner не знайшов цю вразливість.

3.3.4 Вразливість Blind server-side request forgery

Розглянемо вразливість під назвою External service interaction, або як її ще називають, Blind server-side Request Forgery. Експлуатуючи її, можна змусити застосунок виконувати серверні HTTP і HTTPS запити до довільних доменів. В даному випадку було використано сервер Burp Collaborator для генерації корисних навантажень, які відслідковують запити від інших серверів [29]. Тимчасово згенероване посилання http://479rylevgn9q2qxdugalq4vndej878vyjt6ku9.oastify.com/ було передано в запиті у HTTP-заголовку Referer:

Рисунок 3.23 – Спроба експлуатації вразливості

Сервер надсилає HTTP-відповідь з кодом 200 OK, що є першим «дзвіночком», що вразливість є валідною:

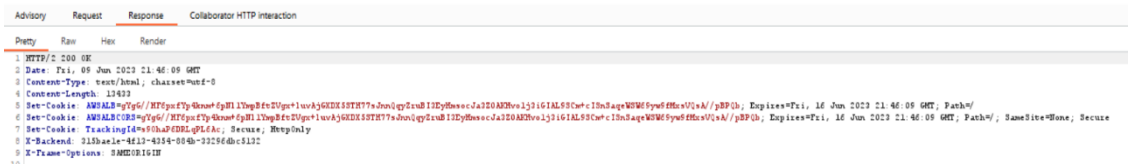

Рисунок 3.24 – Відповідь сервера на зловмисний запит

У вкладці «Collaborator HTTP interaction» міститься доказ валідності даної вразливості, а саме факт звернення тестового вебзастосунку до сервера Burp Collaborator:

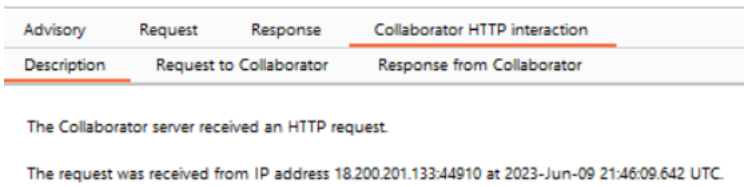

Рисунок 3.25 – Підтвердження знайденої вразливості

У переліку знайдених вразливостей можемо побачити інші схожі екземпляри даної вразливості:

Рисунок 3.26 – Екземпляри вразливості Blind SSRF

Особливість даної вразливості полягає ще й в тому, що вона є наявною на всіх сторінках вебзастосунку, адже корінь проблеми полягає в неправильному налаштуванні перевірок заголовку Referer у бекенді вебзастосунку, тому, так як було підтверджено факт існування вразливості за URL: https://ginandjuice.shop/catalog − всі інші екземпляри даної вразливості автоматично вважаються валідними.

3.3.5 Вразливість SQL injection

Розглянемо вразливість SQL injection, ідентифіковану ПЗ Burp Suite. Вразливості SQL-ін'єкцій виникають, коли контрольовані користувачем дані включаються в SQL-запити до бази даних застосунку у небезпечний спосіб. Зловмисник може ввести втрутитися в структуру такого запиту, щоб вкрасти конфіденційну інформацію користувачів, змінити структуру та наповнення бази даних, видалити з неї будь-які дані або замінити їх своїми [30].

За допомогою SQL-ін'єкцій часто можна здійснити широкий спектр шкідливих атак, включаючи читання або модифікацію критично важливих даних програми, втручання в логіку програми, підвищення привілеїв в базі даних і отримання контролю над сервером бази даних. В даному випадку заявлено, що cookie-параметр TrackingId є вразливим до атак типу SQL injection. Під час тестування застосунку у cookie-параметр TrackingId в рамках HTTP-запиту було додано одну одинарну лапку, у відповідь на що було повернуто загальне повідомлення про помилку. Потім було надіслано дві одинарні лапки, і повідомлення про помилку зникло. Така поведінка вебзастосунку наштовхує на думку про детальний перегляд вмісту повідомлення про помилку у відповіді сервера, а також на роздуми, як програма обробляє інші вхідні дані, щоб переконатися в наявності вразливості.

Спробуємо переконатися у наявності вразливості. На рисунку 6.6 зображено перший запит з однією одинарною лапкою:

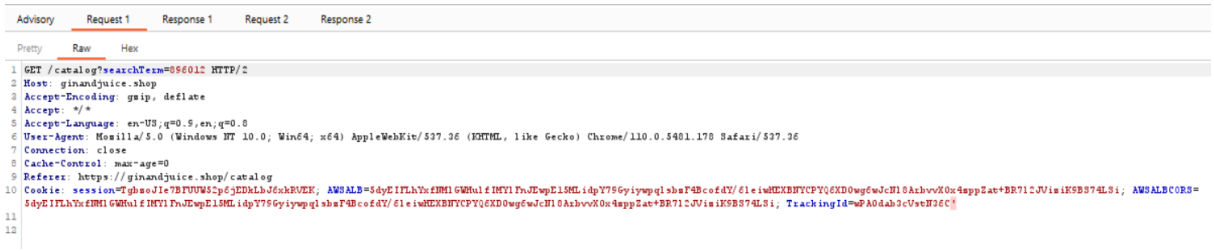

Рисунок 3.27 – Перший HTTP-запит

На рис. 3.28 зображено відповідь застосунку на перший запит:

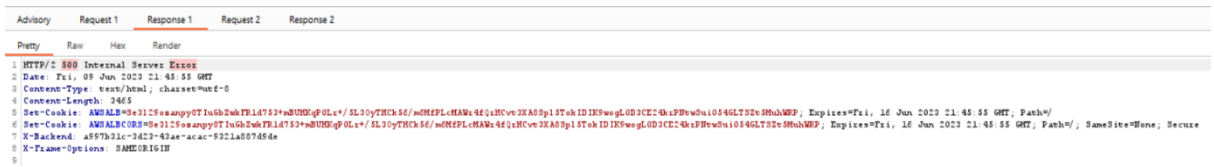

Рисунок 3.28 – Відповідь сервера на перший запит

Відправляємо другий HTTP-запит:

Рисунок 3.29 – Другий HTTP-запит

Отримуємо відповідь застосунку на другий запит:

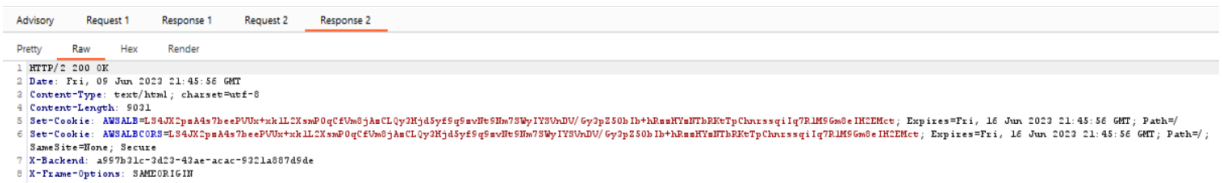

Рисунок 3.30 – Відповідь сервера на другий запит

Проводимо додаткову валідацію вразливості за допомогою утиліти sqlmap. Зберігаємо потенційно вразливий SQL-запит із Burp Suite, натиснувши лівою

кнопкою миші по такому запиту та клацнувши кнопку Save item. Називаємо довільним ім’ям, після чого скануємо цей запит у sqlmap. Проміжні результати виглядають наступним чином:

```
(s) requests:
---
Parameter: Cookie #1* ((custom) HEADER)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause (subquery - comme
nt)
    Payload: session=TgbzoJIe7BFUUW52p6jEDkLbJ6xkRVEK; AWSALB=5dyEIFLhYxfNMlG
WHulfIMYlFnJEwpE15MLidpY79GyiywpqlsbzF4BcofdY/6leiwHEXBNYCPYQ6XD0wg6wJcNl8Arb
vvX0×4zppZat+BR7l2JViziK9BS74LSi; AWSALBCORS=5dyEIFLhYxfNMlGWHulfIMYlFnJEwpE1
5MLidpY79GyiywpqlsbzF4BcofdY/6leiwHEXBNYCPYQ6XD0wg6wJcNl8ArbvvX0×4zppZat+BR7l
2JViziK9BS74LSi; TrackingId=wPA0dab3cVstN36C' AND 8874=(SELECT (CASE WHEN (88
74=8874) THEN 8874 ELSE (SELECT 2745 UNION SELECT 6461) END))-- CBlH

    Type: time-based blind
    Title: MySQL > 5.0.12 OR time-based blind (heavy query)
    Payload: session=TgbzoJIe7BFUUW52p6jEDkLbJ6xkRVEK; AWSALB=5dyEIFLhYxfNMlG
WHulfIMYlFnJEwpE15MLidpY79GyiywpqlsbzF4BcofdY/6leiwHEXBNYCPYQ6XD0wg6wJcNl8Arb
vvX0×4zppZat+BR7l2JViziK9BS74LSi; AWSALBCORS=5dyEIFLhYxfNMlGWHulfIMYlFnJEwpE1
5MLidpY79GyiywpqlsbzF4BcofdY/6leiwHEXBNYCPYQ6XD0wg6wJcNl8ArbvvX0×4zppZat+BR7l
2JViziK9BS74LSi; TrackingId=wPA0dab3cVstN36C' OR 6442=(SELECT COUNT(*) FROM I
NFORMATION_SCHEMA.COLUMNS A, INFORMATION_SCHEMA.COLUMNS B, INFORMATION_SCHEMA
.COLUMNS C)-- njpJ
---
[04:10:58] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL > 5.0.12
[04:10:59] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 610 times
[04:10:59] [INFO] fetched data logged to text files under '/home/kali/.local/
share/sqlmap/output/ginandjuice.shop'
```

Рисунок 3.31 – Додаткова валідація вразливості у sqlmap

Через деякий час, sqlmap видає помилку, бо не може підтвердити наявність вразливості.

```
[04:46:34] [INFO] testing MySQL
[04:46:34] [INFO] confirming MySQL
n
you provided a HTTP Cookie header value, while target URL provides its own cookies within HTTP S
et-Cookie header which intersect with yours. Do you want to merge them in further requests? [Y/n
] y
[04:46:39] [WARNING] the back-end DBMS is not MySQL
[04:46:39] [CRITICAL] sqlmap was not able to fingerprint the back-end database management system
[04:46:39] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 2 times
```

Рисунок 3.32 – Помилка у sqlmap

Таким чином, виходить, що дана вразливість є хибно-позитивною.

Висновки до розділу 3

В розділі 3 було розглянуто сканери вразливостей безпеки як інструментарій для автоматизації та спрощення пошуку вразливостей у вебзастосунках.

У ході проведення порівняльного аналізу сканерів вразливостей було актуалізовано знання з пошуку та експлуатації вразливостей безпеки як на стороні клієнта, так і на стороні сервера, було досліджено роботу програмних засобів Burp Suite та Acunetix Web Vulnerability Scanner на базі тестового середовища у вигляді вебзастосунку – онлайн-магазину уявних товарів.

На основі результатів виконання завдань даного розділу можна зробити висновок, що більш ефективним програмним засобом для тестування вебзастосунків на проникнення є Burp Suite. Даний програмний засіб довів свою ефективність краще, ніж конкуруючий в рамках даного дослідження сканер вразливостей безпеки Acunetix Web Vulnerability Scanner. Не дивлячись на те, що у обох програмних засобів є недоліки, все ж ПЗ Burp Suite має більш значний потенціал для комерційного та некомерційного застосування в рамках проведення тестування вебзастосунків на вразливості безпеки.

4 РЕКОМЕНДАЦІЙНА СИСТЕМА ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАНЬ НА ПРОНИКНЕННЯ ІЗ МЕТОДАМИ ДЛЯ УСУНЕННЯ ВРАЗЛИВОСТЕЙ БЕЗПЕКИ

4.1 Програмна реалізація головного вікна рекомендаційної системи

Програмна реалізація головного вікна була виконана за допомогою технології Windows Forms. Головне вікно складається з поля для введення URL-адреси застосунку, кнопки Ping, яка відправляє пінг-запит на вебсервер, URL якого вказаний у полі для введення, а також віконця, де відображається результат відправки пінг-запиту. Також застосунок містить кнопки Vulnerabilities Mitigation Methods, Scan Configurations, Open Acunetix Scanner та Open Burp Suite.

Перша кнопка відкриває нове вікно, яке містить детальні описи методів усунення вразливостей безпеки на стороні клієнта та на стороні сервера. Друга кнопка відкриває вікно, яке містить конфігурації для проведення тестування на проникнення. Останні дві кнопки запускають програмні засоби Acunetix Web Vulnerability Scanner та Burp Suite відповідно задля того, аби одразу мати можливість запустити тестування застосунку на вразливості безпеки.

Головне вікно програми виглядає наступним чином:

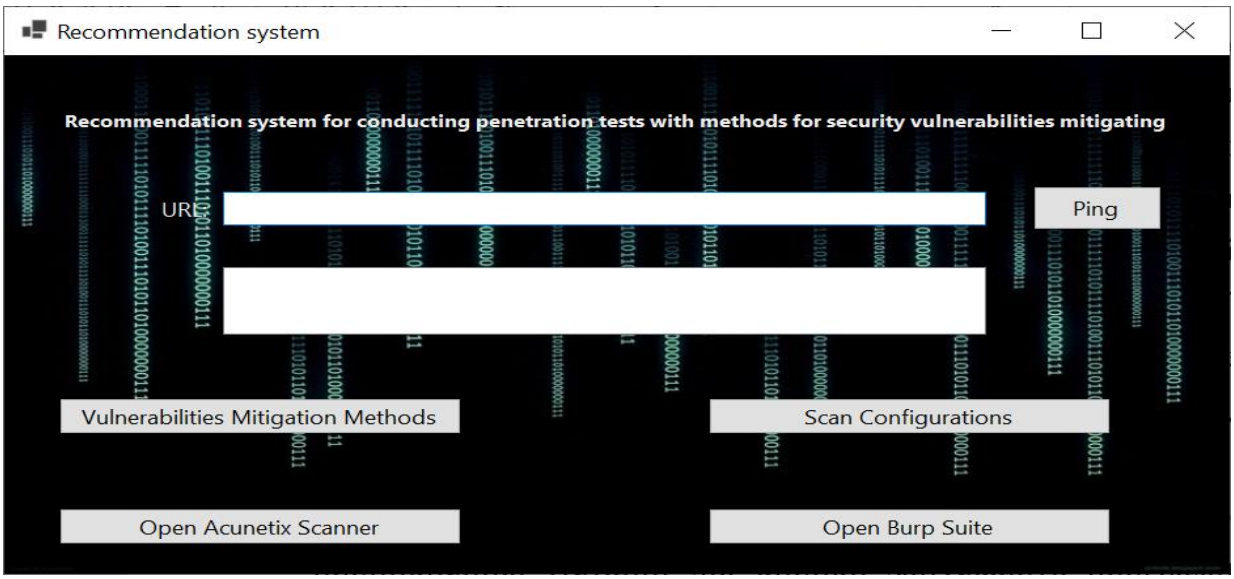

Рисунок 4.1 – Головне вікно рекомендаційної системи

Код для кнопки відкриття вікна із методами усунення вразливостей безпеки вебзастосунків виглядає так:

```
private void OpenVulnerabilitiesButton_Click(object sender, EventArgs e)
      {
         var vulnerabilitiesForm = new VulnerabilitiesForm();
         vulnerabilitiesForm.Show();
      }
```

Код для кнопки відкриття вікна із конфігураціями сканувань виглядає так:

```
private void ScanConfigButton_Click(object sender, EventArgs e)
      {
         var configurationsForm = new ConfigurationsForm();
         configurationsForm.Show();
             }
```

Код для кнопки запуску ПЗ Acunetix Web Vulnerability Scanner виглядає так:

```
private void OpenAcunetixScannerButton_Click(object sender, EventArgs e)
      {
         var myProcess = new Process();
         try
         {
            myProcess.StartInfo.UseShellExecute = true;
            myProcess.StartInfo.FileName = "https://localhost:3443/";
            myProcess.Start();
         }
         catch (Exception ex)
         {
            Console.WriteLine(ex.Message);
         }
      }
```

Код для кнопки запуску ПЗ Burp Suite виглядає так:

```
private void OpenBurpSuiteScannerButton_Click(object sender, EventArgs e)
      {
         var startInfo = new ProcessStartInfo
         {
            FileName = "java.exe",
            Arguments = "-jar C:/Loader.jar"
         };
         Process.Start(startInfo);
      }
   }
            }
```

Код для функціоналу надсилання пінг-запиту та виводу відповіді вебсервера виглядає так:

```
private void PingButton_Click(object sender, EventArgs e)
{
    var url = UrlTextBox.Text;
    string result;
    try
    {
        var pingSender = new Ping();
        var reply = pingSender.Send(url);

        result = reply.Status == IPStatus.Success ? $"Ping successful. Time: {reply.RoundtripTime} ms" : $"Ping failed. Status: {reply.Status}";
    }
    catch (Exception ex)
    {
        result = $"Error: {ex.Message}";
    }
    UrlPingResultTextBox.Text = result;
        }
```

4.2 Програмна реалізація вікна з методами усунення вразливостей безпеки

Програмна реалізація вікна з методами усунення вразливостей безпеки була виконана за допомогою технології Windows Forms. Вікно складається з переліку назв та типів вразливостей, методів усунення вразливостей, поля для пошуку вразливості за назвою у загальному списку, кнопки Find та фільтру за типами вразливостей. Вікно має такий загальний вигляд:

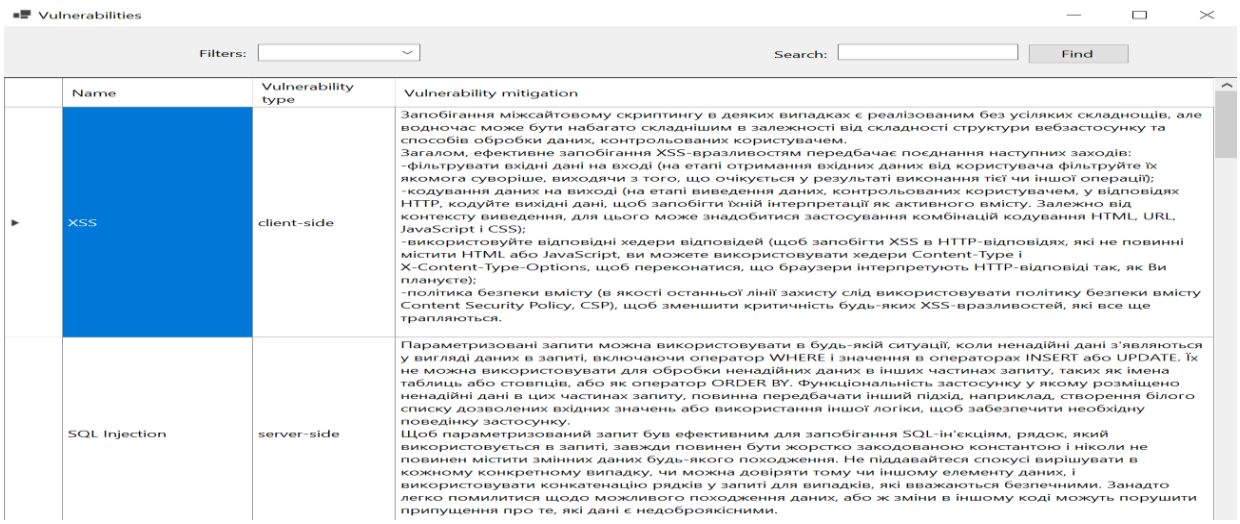

Рисунок 4.2 – Методи усунення вразливостей безпеки

Код програмної реалізації даного компоненту рекомендаційної системи міститься в додатку А.

4.3 Програмна реалізація вікна з конфігураціями проведення тестування на проникнення

Програмна реалізація вікна з конфігураціями проведення тестування на проникнення була виконана за допомогою технології Windows Forms. Вікно складається з переліку типів вебзастосунків, приблизної к-сті сторінок у них, прикладну сферу вебзастосунка, конфігурації для кравлінгу та аудиту, поля для пошуку конфігурації за типом застосунку, кнопки Find та фільтру за к-стю сторінок вебзастосунка.

Код програмної реалізації даного компоненту рекомендаційної системи міститься в додатку Б.

Висновки до розділу 4

В цьому розділі було повністю описано програмну реалізацію рекомендаційної системи для тестування вебзастосунків на проникнення із методами усунення вразливостей безпеки. Метою написання даного розділу було створення рекомендаційної системи задля спрощення процесу проведення тестувань на проникнення та спрощення розуміння методів усунення виявлених під час тестування на проникнення вразливостей, що й було досягнено.

Опис програмної реалізації рекомендаційної системи є важливою частиною, адже це надає розуміння, як саме працює рекомендаційна система, які завдання та за яких умов може виконувати. Демонстрація програмної реалізації зумовлює точніше враження про систему.

Рекомендаційна система є допоміжним інструментом для внеску у дослідження проблем і завдань бакалаврської кваліфікаційної роботи. За результатами розробки можна з’ясувати, що поставлене завдання виконано, а сама реалізація може бути повністю або частково використана для модернізації

даного рішення задля розвитку у щось більш комплексне з метою подолання сучасних викликів кібербезпеки.

ВИСНОВКИ

В ході виконання бакалаврської кваліфікаційної роботи було досягнуто всіх поставлених цілей: досліджено наявні методології тестування вебзастосунків на проникнення, досліджено актуальність проблеми кібератак та інцидентів безпеки в Україні до початку повномасштабної війни та після її початку, проаналізовано модель дискреційного розмежування доступу Take-Grant та її практичне застосування, проведено порівняльний аналіз сканерів вебзастосунків на вразливості безпеки та представлено рекомендаційну систему для проведення тестування на проникнення із методами усунення вразливостей, знайдених у результаті такого тестування.

Методології тестування на проникнення є актуальними та сучасними, матеріали цих методологій було застосовано у процесі проведення тестування на проникнення тестового вебзастосунку.

Проблема кібератак в Україні стала ще більш нагальною після початку повномасштабного вторгнення, потребує серйозної уваги з боку держави та спеціалістів у галузі кібербезпеки. Захист вебзастосунків в приватному та державному секторах України був актуальним і раніше, але став надзвичайно актуальним станом на зараз з урахуванням зовнішніх викликів та загроз. Фахівцям з кібербезпеки слід приділяти особливу увагу безпеці вебзастосунків сьогодні та в майбутньому, імплементуючи кращі практики безпеки із визнаних спільнотою методологій.

Було досліджено модель дискреційного розмежування доступу Take-Grant. Крізь роки свого існування, модель не втрачає своєї актуальності, є суб’єктивно нескладною для розуміння та має практичне застосування у реальному житті при проектуванні, створенні та керуванні вебзастосунками.

Було проведено порівняльний аналіз сканерів вразливостей безпеки вебзастосунків Burp Suite та Acunetix Web Vulnerability Scanner. У результаті проведення такого аналізу було виявлено, що ПЗ Burp Suite є більш ефективним у якості допоміжного інструмента при проведенні тестування на проникнення.

Було представлено рекомендаційну систему для тестування вебзастосунків на проникнення. Дана рекомендаційна система також містить методи усунення вразливостей безпеки – як вразливостей на стороні клієнта, так і вразливостей на стороні сервера. Такий програмний продукт має значну цінність, адже містить конфігурації для проведення тестування на проникнення для програмних засобів Burp Suite та Acunetix Web Vulnerability Scanner, які можна застосувати для аудиту безпеки будь-якого вебзастосунку, та економить час на пошуки коректних налаштувань сканерів вразливостей безпеки перед початком тестування на проникнення.

Мета даної роботи полягала у дослідженні проблеми кібератак в Україні на державний та приватний сектори до та після початку повномасштабного вторгнення, виявленні вразливостей безпеки у вебзастосунках задля демонстрації важливості створення, конфігурування та адміністрування безпечних вебзастосунків, створенні рекомендаційної системи для проведення тестувань на проникнення із методами усунення вразливостей безпеки, знайдених внаслідок проведення таких тестувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Пентести за допомогою сканерів – за чи проти?: вебсайт. URL: https://10guards.com/ua/articles/penetration-testers-with-or-without-vulnerability-scanners/ (дата звернення 15.05.2023).
2. Dynamic application security testing: вебсайт. URL: https://en.wikipedia.org/wiki/Dynamic_application_security_testing (дата звернення 15.05.2023).
3. OSSTMM 3. The Open Source Security Testing Methodology Manual. Contemporary Security Testing and Analysis: вебсайт. URL: https://www.isecom.org/OSSTMM.3.pdf (дата звернення 15.05.2023).
4. Technical Guide to Information Security Testing and Assessment: вебсайт. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf (дата звернення 15.05.2023).
5. OWASP Top Ten: вебсайт. URL: https://owasp.org/www-project-top-ten/ (дата звернення 15.05.2023).
6. CWE Top 25: These are the most dangerous software weaknesses of 2022: вебсайт. URL: https://portswigger.net/daily-swig/cwe-top-25-these-are-the-most-dangerous-software-weaknesses-of-2022 (дата звернення 15.05.2023).
7. PTES Technical Guidelines: вебсайт. URL: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines (дата звернення 15.05.2023).
8. Information Systems Security Assessment Framework (ISSAF) Draft 0.2.1: вебсайт. URL: https://untrustednetwork.net/files/issaf0.2.1.pdf (дата звернення 15.05.2023).
9. Російсько-українська кібервійна: вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%BE-%D1%83%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%81%D1%8C%D0%BA%D0%B0_%D0%BA%D1%96%D0%B1%D0%B5%D1%80%D0%B2%D1%96%D0%B9%D0%BD%D0%B0 (дата звернення 15.05.2023).

10. Кібератаки на українські державні сайти (2022): вебсайт. URL: https://uk.wikipedia.org/wiki/%D0%9A%D1%96%D0%B1%D0%B5%D1%80%D0%B0%D1%82%D0%B0%D0%BA%D0%B8_%D0%BD%D0%B0_%D1%83%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D1%81%D1%8C%D0%BA%D1%96_%D0%B4%D0%B5%D1%80%D0%B6%D0%B0%D0%B2%D0%BD%D1%96_%D1%81%D0%B0%D0%B9%D1%82%D0%B8_(2022) (дата звернення 15.05.2023).
11. Найбільші кібератаки проти України з 2014 року. Інфографіка: вебсайт. URL: https://nv.ua/ukr/ukraine/events/najbilshi-kiberataki-proti-ukrajini-z-2014-roku-infografika-1438924.html (дата звернення 15.05.2023).
12. Russian APT and Ransomware Groups: Vulnerabilities and Threat Actors Who Exploit Them: вебсайт. URL: https://flashpoint.io/blog/vulnerabilities-exploited-by-russian-apts-ransomware-groups/ (дата звернення 15.05.2023).
13. The Epic Turla Operation: вебсайт. URL: https://securelist.com/the-epic-turla-operation/65545/ (дата звернення 16.05.2023).
14. Як заражалася Україна: хронологія найбільшої в історії кібератаки: вебсайт. URL: https://www.epravda.com.ua/publications/2017/07/20/627290/ (дата звернення 16.05.2023).
15. Розбір | «За масштабами атак – це Росія». Внутрішній фронт: як Кремль розхитує Україну: вебсайт. URL: https://www.liga.net/ua/politics/articles/po-masshtabam-atak-eto-rossiya-vnutrenniy-front-kak-kreml-raskachivaet-ukrainu (дата звернення 15.05.2023).
16. СБУ відзвітувала про нейтралізацію понад 4,5 тисячі кібератак з початку року: вебсайт. URL: https://delo.ua/telecom/sbu-vidzvituvala-pro-neitralizaciyu-ponad-45-tisyaci-kiberatak-z-pocatku-roku-408759/ (дата звернення 15.05.2023).
17. ESET Research: Russian APT groups, including Sandworm, continue their attacks against Ukraine with wipers and ransomware: вебсайт. URL: https://www.eset.com/int/about/newsroom/press-releases/research/eset-research-

russian-apt-groups-including-sandworm-continue-their-attacks-against-ukraine-with-wipe/ (дата звернення 16.05.2023).

18. Web Security Academy. Learning path: вебсайт. URL: https://portswigger.net/web-security/learning-path (дата звернення 16.05.2023).
19. Що таке керування вразливостями?: вебсайт. URL: https://www.microsoft.com/uk-ua/security/business/security-101/what-is-vulnerability-management (дата звернення 15.05.2023).
20. William L. Simon, Donna Beech Ghost In The Wires: My Adventures as the World's Most Wanted Hacker. Back Bay Books; 1st edition April 24, 2012. 448 p.
21. Best Practices: Use of Web Application Firewalls: вебсайт. URL: https://owasp.org/www-pdf-archive/Best_Practices_Guide_WAF_v104.en.pdf (дата звернення 16.05.2023).
22. Configuring Settings – Introduction: вебсайт. URL: https://www.acunetix.com/support/docs/wvs/acunetix-settings/ (дата звернення 16.05.2023).
23. Оцінка ефективності сканерів безпеки веб-додатків: вебсайт. URL: https://periodicals.karazin.ua/cscs/article/view/18180/16594 (дата звернення 15.05.2023).
24. Web Application Security Attacks and Mitigation techniques: вебсайт. URL: https://systemweakness.com/web-application-security-attacks-and-mitigation-techniques-1dfcc40200ff (дата звернення 16.05.2023).
25. Client-Side Attacks: What They Are and How to Prevent Them: вебсайт. URL: https://cheq.ai/blog/client-side-website-data-theft-javascript/ (дата звернення 16.05.2023).
26. Adam Shostack Threat Modeling: Designing for Security. Wiley; 1st edition February 17, 2014. 624 p.
27. Mary Aiken The Cyber Effect: An Expert in Cyberpsychology Explains How Technology Is Shaping Our Children, Our Behavior, and Our Values--and What We Can Do About It. Random House; 1st edition June 27, 2017. 400 p.

28. Jon Erickson Hacking: The Art of Exploitation. No Starch Press; 2nd edition February 4, 2008. 488 p.
29. Stuart McClure, George Kurtz, Joel Scambray Hacking Exposed 7: Network Security Secrets and Solutions. McGraw Hill; 7th edition August 1, 2012. 768 p.
30. SQL Injection Cheet Sheet: вебсайт. URL: https://www.invicti.com/blog/web-security/sql-injection-cheat-sheet/ (дата звернення 16.05.2023).

ДОДАТОК А
Файл VulnerabilitiesForm.cs

```
public partial class VulnerabilitiesForm : Form
  {
    public VulnerabilitiesForm()
    {
      InitializeComponent();
      LoadDataGrid();
      VulnerabilityDataGridView.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
      VulnerabilityDataGridView.Columns[2].DefaultCellStyle.WrapMode = DataGridViewTriState.True;
      FiltersComboBox.Items.AddRange(new object[] { "server-side", "client-side", "all" });
    }
    private void SearchButton_Click(object sender, EventArgs e)
    {
      LoadDataGrid();
      var rowsCollection = VulnerabilityDataGridView.Rows;
      var searchedRowsCollection = new List<DataGridViewRow>();
      searchedRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
row.Cells[0].Value.ToString()!.ToLower().Contains(SearchBarTextBox.Text.ToLower())));

      VulnerabilityDataGridView.Rows.Clear();
      foreach (var row in searchedRowsCollection)
      {
        VulnerabilityDataGridView.Rows.Add(row);
      }
    }

    private void FiltersComboBox_SelectedIndexChanged(object sender, EventArgs e)
    {
      LoadDataGrid();
      var rowsCollection = VulnerabilityDataGridView.Rows;
      var filteredRowsCollection = new List<DataGridViewRow>();
      switch (FiltersComboBox.SelectedIndex)
      {
        case 0:
          {                    filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
row.Cells[1].Value.ToString()!.Contains("server-side")));
            break;
          }
        case 1:
          {
       filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
row.Cells[1].Value.ToString()!.Contains("client-side")));

            break;
          }
        default:
          LoadDataGrid();
          return;
      }
      VulnerabilityDataGridView.Rows.Clear();
      foreach (var row in filteredRowsCollection)
      {
        VulnerabilityDataGridView.Rows.Add(row);
      }
    }
    private void LoadDataGrid()
    {
      VulnerabilityDataGridView.Rows.Clear();
      VulnerabilityDataGridView.Rows.Add("XSS", "client-side", "Запобігання міжсайтовому скриптингу в деяких випадках є реалізованим без усіляких складнощів, але водночас може бути набагато складнішим в залежності від складності структури вебзастосунку та способів обробки даних, контрольованих користувачем.\r\nЗагалом, ефективне запобігання XSS-вразливостям передбачає поєднання наступних заходів:\r\n-\tфільтрувати вхідні дані на вході (на етапі отримання вхідних даних від користувача фільтруйте їх якомога суворіше, виходячи з того, що очікується у результаті виконання тієї чи іншої операції);\r\n-\tкодування даних на виході (на етапі виведення даних, контрольованих користувачем, у відповідях HTTP, кодуйте вихідні дані,
```

щоб запобігти їхній інтерпретації як активного вмісту. Залежно від контексту виведення, для цього може знадобитися застосування комбінацій кодування HTML, URL, JavaScript і CSS);\r\n-\tвикористовуйте відповідні хедери відповідей (щоб запобігти XSS в HTTP-відповідях, які не повинні містити HTML або JavaScript, ви можете використовувати хедери Content-Type і X-Content-Type-Options, щоб переконатися, що браузери інтерпретують HTTP-відповіді так, як Ви плануєте);\r\n-\tполітика безпеки вмісту (в якості останньої лінії захисту слід використовувати політику безпеки вмісту Content Security Policy, CSP), щоб зменшити критичність будь-яких XSS-вразливостей, які все ще трапляються.\r\n");

VulnerabilityDataGridView.Rows.Add("SQL Injection", "server-side", "Параметризовані запити можна використовувати в будь-якій ситуації, коли ненадійні дані з'являються у вигляді даних в запиті, включаючи оператор WHERE і значення в операторах INSERT або UPDATE. Їх не можна використовувати для обробки ненадійних даних в інших частинах запиту, таких як імена таблиць або стовпців, або як оператор ORDER BY. Функціональність застосунку у якому розміщено ненадійні дані в цих частинах запиту, повинна передбачати інший підхід, наприклад, створення білого списку дозволених вхідних значень або використання іншої логіки, щоб забезпечити необхідну поведінку застосунку.\r\nЩоб параметризований запит був ефективним для запобігання SQL-ін'єкціям, рядок, який використовується в запиті, завжди повинен бути жорстко закодованою константою і ніколи не повинен містити змінних даних будь-якого походження. Не піддавайтеся спокусі вирішувати в кожному конкретному випадку, чи можна довіряти тому чи іншому елементу даних, і використовувати конкатенацію рядків у запиті для випадків, які вважаються безпечними. Занадто легко помилитися щодо можливого походження даних, або ж зміни в іншому коді можуть порушити припущення про те, які дані є недоброякісними.\r\n");

VulnerabilityDataGridView.Rows.Add("CSRF", "client-side", "Найнадійнішим способом захисту від CSRF-атак є включення токену CSRF у відповідні запити. Токен повинен відповідати наступним критеріям:\r\n-\tбути непередбачуваним з високою ентропією, як і для сесійних токенів в цілому;\r\n-\tбути прив'язаним до сесії користувача;\r\n-\tсуворо валідуватися в кожному випадку перед виконанням відповідної дії у вебзастосунку.\r\nНа додачу до впровадження надійної перевірки токенів CSRF, рекомендовано явно встановлювати флаги SameSite для кожного файлу cookie. Таким чином, з’явиться можливість точно контролювати, в яких контекстах будуть використовуватися файли cookie, незалежно від браузера.\r\nНавіть якщо всі браузери врешті-решт приймуть політику \"Lax-by-default\", вона не підходить для кожного файлу cookie, і її легше обійти, ніж обмеження типу Strict. У той же час, невідповідність між різними браузерами також означає, що лише частина ваших користувачів отримає вигоду від будь-якого захисту SameSite.\r\nВ ідеалі, за замовчуванням слід використовувати політику Strict, а потім знизити її до Lax, тільки якщо у вас є на це вагома причина. Ніколи не варто вимикати обмеження SameSite за допомогою встановлення значення None, якщо немає усвідомлення наслідків для безпеки.\r\nНедопущення cross-origin same-site атак. Правильно налаштовані обмеження SameSite забезпечують хороший захист від міжсайтових атак, важливо розуміти, що вони абсолютно безсилі проти cross-origin same-site атак. Якщо це можливо - рекомендовано ізолювати небезпечний вміст від будь-якого чутливого функціоналу або даних, наприклад, завантажені користувачем файли, на окремому сайті. Під час тестування сайту варто обов'язково провести ретельний аудит усіх доступних attack surfaces, що належать до цього ж сайту, включно з будь-якими його дочірніми доменами.\r\n");

VulnerabilityDataGridView.Rows.Add("CORS", "client-side", "Вразливості CORS виникають в першу чергу через неправильні конфігурації сервера. Ефективне запобігання експлуатації CORS-вразливостей включає в себе такі кроки:\r\n-\tправильна конфігурація cross-site запитів: якщо вебзастосунок містить конфіденційну інформацію, його походження має бути правильно вказано в заголовку Access-Control-Allow-Origin;\r\n-\tдозвіл тільки довірених сайтів: це може здатися очевидним, але джерелом, зазначеним у заголовку Access-Control-Allow-Origin, повинні бути лише довірені сайти. Зокрема, динамічне відображення походження з перехресних запитів без перевірки може бути використане зловмисниками, і цього слід уникати;\r\n-\tуникання значення null в білому списку: уникайте використання заголовка Access-Control-Allow-Origin: null. Хедери CORS повинні бути належним чином визначені щодо довіреного походження для внутрішніх та публічних серверів;\r\n-\tуникання підстановочних знаків (*) у внутрішніх мережах: уникайте використання підстановочних знаків у внутрішніх мережах. Для захисту внутрішніх ресурсів недостатньо лише надійної конфігурації мережі, коли внутрішні браузери можуть отримати доступ до ненадійних зовнішніх доменів;\r\n-\tвпровадження політик CORS разом із політиками безпеки на стороні сервера: CORS визначає поведінку браузера і ніколи не замінить захист конфіденційних даних на стороні сервера, адже зловмисник може безпосередньо підробити запит з будь-якого надійного джерела. Тому вебзастосунки повинні продовжувати застосовувати засоби захисту конфіденційних даних на стороні сервера, такі як автентифікація та управління сеансами, на додаток до належним чином налаштованого CORS.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Clickjacking", "client-side", "Найпоширенішим механізмом запобігання на стороні браузера довгий час були скрипти перехоплення фреймів (frame busting scripts). Згодом стало зрозуміло, що зловиснику часто дуже легко обійти цей захист. У відповідь на це були розроблені серверні протоколи, які обмежують використання браузером iframe і захищають від перехоплення кліків.\r\nКлікджекінг - це вид атаки на стороні браузера, і його успіх або неуспіх залежить від функціональності браузера та відповідності до існуючих вебстандартів і best practices безпеки. Захист від клікджекінгу на стороні сервера забезпечується шляхом визначення обмежень на використання таких компонентів у Javascript, як iframe. Однак реалізація захисту залежить від дотримання та виконання цих обмежень браузером. Опираючись на вищесказане, варто зазначити, шо все ж найефективнішими механізмами захисту від клікджекінгу на стороні сервера є наявність правильно сконфігурованих HTTP-заголовків X-Frame-Options та Content Security Policy.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Command injection", "server-side", "Безумовно, найефективнішим способом запобігти вразливостям, пов'язаним з ін'єкціями команд ОС - це ніколи не звертатися до команд ОС з коду на рівні застосунку. Практично в кожному випадку існують альтернативні способи реалізації необхідної функціональності за допомогою безпечних API платформ.\r\nЯкщо виклик команд операційної системи за допомогою користувацького вводу вважається неминучим, необхідно імплементувати ретельну перевірку користувацького вводу. Деякі методи ефективної валідації даних включають:\r\n-\tперевірка за білим списком дозволених значень;\r\n-\tперевірка того, що введене значення є числом;\r\n-\tперевірка того, що вхідні дані містять лише алфавітно-цифрові символи, без інших синтаксичних знаків або пробілів.\r\nНіколи не намагайтеся очистити вхідні дані за допомогою екранування метасимволів оболонки. На практиці, це надто вразливий спосіб, який може призвести до помилок і який може бути обійдений досвідченим зловмисником.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Information disclosure", "server-side", "Повністю запобігти розкриттю інформації складно через величезну різноманітність способів, якими це може відбуватися. Однак є кілька загальних найкращих практик, яких варто дотримуватися, щоб мінімізувати ризик виникнення information disclosure:\r\n-\tпереконайтеся, що всі, хто бере участь у створенні вебзастосунку, повністю усвідомлюють, яка інформація, яку він обробляє або зберігає, вважається конфіденційною. Іноді інформація, яка здається безсенсовною, може бути набагато кориснішою для зловмисника, ніж люди можуть собі уявити. Висвітлення небезпек розкриття інформації може допомогти забезпечити більш безпечне поводження з конфіденційною інформацією у вашій організації в цілому;\r\n-\tпроводьте аудит будь-якого коду на предмет потенційного розголошення інформації в рамках процесів контролю якості або збірки. Автоматизувати деякі пов'язані з цим завдання, наприклад, видалення коментарів розробників, має бути відносно легко;\r\n-\tвикористовуйте загальні повідомлення про помилки замість конкретних якомога частіше. Не давайте зловмисникам підказки про поведінку вебзастосунку в тих чи інших сценаріях без необхідності;\r\n-\tперевірте, чи всі функції діагностики (debug mode) вимкнені у виробничому середовищі застосунку;\r\n-\tпереконайтеся, що ви повністю розумієте наслідки для безпеки будь-якої сторонньої технології, яку ви впроваджуєте в програмний продукт;\r\n-\tзнайдіть час, щоб вивчити та вимкнути всі функції та налаштування вебзастосунку, які наразі не потрібні.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Path traversal", "server-side", "Найефективніший спосіб запобігти вразливостям обходу шляху до файлу - це взагалі уникати передачі даних, що вводяться користувачем, до API файлової системи. Багато функцій застосунку, які роблять це, можна переписати, щоб забезпечити ту ж саму поведінку у безпечніший спосіб.\r\nЯкщо передача введених користувачем даних до API файлової системи вважається неминучою, то для запобігання атакам даного типу слід використовувати два рівні захисту:\r\n-\tзастосунок повинен перевіряти введені користувачем дані перед їх обробкою. В ідеалі, перевірка повинна порівнюватись з білим списком дозволених значень. Якщо таку перевірку неможливо реалізувати за якихось причин, то повинен існувати метод перевірки, при якому вхідні дані будуть містити лише дозволений вміст, наприклад, суто алфавітно-цифрові символи;\r\n-\tпісля перевірки вхідних даних вебзастосунок повинен додати їх до базового каталогу та використати API файлової системи для канонізації шляху. Програма повинна перевірити, що канонізований шлях починається з очікуваного базового каталогу.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("DOM-based vulnerabilities", "client-side", "Не існує якогось одного конкретного методу, який б повністю усував загрозу DOM-атак. Однак, загалом кажучи, найефективніший спосіб уникнути DOM-вразливостей - не дозволяти даним з будь-якого ненадійного джерела (untrusted source) динамічно змінювати значення, що передаються до будь-якого поглинача (sink).\r\nЯкщо бажана функціональність вебзастосунку свідчить про те, що ризики атак на DOM є неминучими, то засоби захисту повинні бути реалізовані в коді на стороні клієнта. У багатьох випадках відповідні дані можна перевірити на основі білого списку, дозволивши лише той вміст, про який відомо, що він є безпечним. В інших випадках необхідно перевіряти дані або кодувати їх. Це може бути складним завданням і, залежно від контексту, в який потрібно вставити дані, може включати комбінацію екранування JavaScript, кодування HTML і кодування URL-адрес у відповідній послідовності.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("WebSocket vulnerabilities", "client-side", "Щоб мінімізувати ризик вразливостей безпеки, що виникають при використанні WebSockets, варто дотримуватися наступних рекомендацій:\r\n-\tвикористовуйте протокол wss:// (WebSockets over TLS) замість ws://;\r\n-\tкодуйте URL-посилання ендпойнтів WebSockets і, звичайно, не включайте в це URL-посилання дані, контрольовані користувачем;\r\n-\tзахистіть WebSocket-хендшейки від CSRF, щоб уникнути міжсайтових вразливостей перехоплення контролю над WebSockets;.\r\n-\tрозглядайте дані, отримані через WebSocket, як ненадійні в обох напрямках – відправки та отримання;\r\n-\tбезпечно обробляйте дані як на стороні сервера, так і на стороні клієнта, щоб запобігти вразливостям на основі ін’єкцій, таким як SQL-ін'єкції та міжсайтовий скриптинг.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Arbitrary file upload", "server-side", "Дозвіл користувачам завантажувати файли є звичайною справою і не повинен бути небезпечним, якщо вживаються належні заходи безпеки. Загалом, найефективніший спосіб захистити власні вебзастосунки від вразливостей такого виду - застосувати всі наведені нижче практики:\r\n-\tперевірка розширення файлу за білим списком дозволених розширень, а не за чорним списком заборонених. Набагато легше зрозуміти, які розширення ви можете дозволити, ніж здогадатися, які розширення може спробувати завантажити зловмисник;\r\n-\tперевірка імені файлу на відсутність підрядків, які можуть бути інтерпретовані як каталог або послідовність обходу (../);\r\n-\tперейменування завантажених файлів задля уникнення колізій, які можуть призвести до перезапису існуючих файлів;\r\n-\tзаборона завантаження файлів до постійної файлової системи вебзастосунку, доки вони не будуть повністю перевірені;\r\n-\tвикористання вже існуючих фреймворків для попередньої обробки завантажених файлів замість використання власноруч написаних власних механізмів валідації.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("XXE injection", "server-side", "Практично всі XXE-вразливості виникають через те, що бібліотека синтаксичного аналізу XML у застосунку підтримує потенційно небезпечні функції XML, які насправді не планують використовувати. Найпростіший і найефективніший спосіб запобігти XXE-атакам - вимкнути ці функції.\r\nЯк правило, достатньо вимкнути роздільну здатність зовнішніх об'єктів і відключити підтримку XInclude. Зазвичай це можна зробити за допомогою параметрів конфігурації або програмного перевизначення поведінки за замовчуванням. Зверніться до документації до вашої бібліотеки синтаксичного аналізу XML або API для отримання детальної інформації про те, як вимкнути непотрібні функції.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Broken access control", "server-side", "Вразливостям контролю доступу зазвичай можна запобігти, застосовуючи підхід до захисту в глибину і дотримуючись наступних принципів:\r\n-\tякщо ресурс не призначений для публічного доступу, забороняйте доступ до нього за замовчуванням;\r\n-\tвикористовуйте єдиний механізм контролю доступу для всього застосунку;\r\n-\tна рівні коду зробіть обов'язковим для розробників декларування дозволеного доступу до кожного ресурсу та забороняйте доступ за замовчуванням;\r\n-\tретельно перевіряйте та тестуйте засоби контролю доступу, щоб переконатися, що вони працюють належним чином.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Authentication vulnerabilities", "server-side", "Аутентифікація - це складна тема, і в ній, на жаль, дуже легко закрадаються вразливості. Описати всі можливі заходи, які можна вжити для захисту власних вебзастосунків, очевидно, неможливо. Однак є кілька загальних принципів, яких завжди слід дотримуватися:\r\n-\tбудьте обережні з обліковими даними користувачів: навіть найнадійніші механізми автентифікації будуть неефективними, якщо ви

ненавмисно розкриєте зловмиснику дійсний набір облікових даних для входу в систему. Само собою зрозуміло, що ви ніколи не повинні надсилати будь-які дані для входу через незашифровані з'єднання. Хоча ви могли впровадити HTTPS для своїх запитів на вхід, переконайтеся, що ви дотримуєтеся цього, перенаправляючи будь-які спроби HTTP-запитів на HTTPS;\r\nВи також повинні провести аудит вашого вебсайту, щоб переконатися, що жодне ім'я користувача або адреса електронної пошти не розголошуються через загальнодоступні профілі або не відображаються, наприклад, у відповідях HTTP.\r\n-\tНе покладайтеся на користувачів у питаннях безпеки: суворі заходи автентифікації часто вимагають додаткових зусиль від ваших користувачів. Людська природа робить майже неминучим те, що деякі користувачі знайдуть способи позбавити себе цих зусиль. Тому вам потрібно забезпечити безпечну поведінку скрізь, де це можливо;\r\nНайочевиднішим прикладом є впровадження ефективної політики паролів. Деякі з найбільш традиційних політик не працюють, тому що люди вписують в них свої власні передбачувані паролі. Замість цього може бути більш ефективним впровадити просту перевірку паролів, яка дозволяє користувачам експериментувати з паролями і надає зворотній зв'язок про їхню надійність в реальному часі. Популярним прикладом є бібліотека JavaScript zxcvbn, розроблена Dropbox. Дозволяючи використовувати лише ті паролі, які отримали високу оцінку програми перевірки паролів, ви можете забезпечити використання безпечних паролів більш ефективно, ніж за допомогою традиційних політик.\r\nЗловмиснику значно легше зламати ваші механізми автентифікації, якщо ви виявите, що користувач існує в системі. Існують навіть певні ситуації, коли через природу вебсайту знання про те, що певна особа має обліковий запис, саме по собі є конфіденційною інформацією.\r\nНезалежно від того, чи дійсне ім'я користувача, яке намагаються ввести, важливо використовувати ідентичні, загальні повідомлення про помилки і переконатися, що вони дійсно ідентичні. Ви завжди повинні повертати один і той же HTTP-код статусу з кожним запитом на вхід і, нарешті, зробити час відповіді в різних сценаріях максимально нерозрізненим.\r\nЗ огляду на те, наскільки простою може бути побудова атаки грубого перебору, дуже важливо вжити заходів для запобігання або, принаймні, припинення будь-яких спроб підбору логінів грубим перебором.\r\n\r\n");

VulnerabilityDataGridView.Rows.Add("Business logic vulnerabilities", "server-side", "Методами усунення вразливостей бізнес-логіки є:\r\n-\tрозуміння того, що розробники та тестувальники розуміють домен (галузь), який (яку) обслуговує застосунок;\r\n-\tуникання неявних припущень про поведінку користувачів або поведінку інших частин застосунку. Ви повинні визначити, які припущення ви зробили про стан вебзастосунку на стороні сервера, і реалізувати необхідну логіку для перевірки того, що ці припущення виконуються. Це включає в себе переконання в тому, що значення будь-яких вхідних даних є в межах заздалегідь зазначених значень.\r\nТакож важливо переконатися, що і розробники, і тестувальники повністю розуміють ці припущення і те, як застосунок повинен реагувати в різних сценаріях. Це може допомогти команді виявити недоліки логіки якомога раніше. Щоб полегшити це, команда розробників повинна дотримуватися наступних найкращих практик, де це можливо:\r\n-\tведіть чіткі проектні документи і потоки даних для всіх транзакцій і робочих процесів, відзначаючи всі припущення, які робляться на кожному етапі;\r\n-\tпишіть код якомога зрозуміліше: якщо важко зрозуміти, що має відбуватися у коді застосунку, буде важко виявити будь-які логічні помилки. В ідеалі, добре написаний код не повинен потребувати документації, щоб його зрозуміти. У неминуче складних випадках створення чіткої документації має вирішальне значення для того, щоб інші розробники та тестувальники знали, які припущення були зроблені і яка саме поведінка є очікуваною;\r\n-\tзверніть увагу на будь-які посилання на інший код, який використовує кожен компонент застосунку: подумайте про побічні ефекти цих залежностей, якщо зловмисник зможе маніпулювати ними у незвичний спосіб.\r\nЧерез відносно унікальну природу багатьох логічних недоліків, легко відмахнутися від них як від одноразової помилки, спричиненої людським фактором, і рухатися далі. Однак, ці недоліки часто є результатом реалізації недосконалих практик безпеки на початкових етапах створення вебзастосунку. Аналіз того, чому виникла логічна помилка і як вона залишалася непоміченою, може допомогти виявити слабкі місця у процесах. Внісши незначні корективи, ви можете підвищити ймовірність того, що подібні помилки будуть виявлені на більш ранніх етапах розробки або ж не будуть виникати взагалі.\r\n\r\n");

}

ДОДАТОК Б
Файл ConfigurationsForm.cs

```
public partial class ConfigurationsForm : Form
        {
            public ConfigurationsForm()
            {
                InitializeComponent();
                ConfigurationsDataGridView.Columns[2].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                ConfigurationsDataGridView.Columns[3].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                ConfigurationsDataGridView.Columns[4].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                ConfigurationsDataGridView.Columns[2].DefaultCellStyle.WrapMode = DataGridViewTriState.True;
                ConfigurationsDataGridView.Columns[3].DefaultCellStyle.WrapMode = DataGridViewTriState.True;
                ConfigurationsDataGridView.Columns[4].DefaultCellStyle.WrapMode = DataGridViewTriState.True;
                LoadDataGrid();
                FiltersComboBox.Items.AddRange(new object[] { "1-10", "11-50", "51-200", "201+", "all" });
            }
            private void FiltersComboBox_SelectedIndexChanged(object sender, EventArgs e)
            {
                LoadDataGrid();
                var rowsCollection = ConfigurationsDataGridView.Rows;
                var filteredRowsCollection = new List<DataGridViewRow>();
                switch (FiltersComboBox.SelectedIndex)
                {
                    case 0:
                        {
filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
                            {
                                var input = row.Cells[1].Value.ToString();
                                var list = input?.Split('-').Select(int.Parse).ToList();
                                if (list!.Count == 1)
                                {
                                    return 1 <= list[0] && list[0] <= 10;
                                }
                                return 1 <= list[0] && list[0] <= 10 || 1 <= list[1] && list[1] <= 10;
                            }));
                            break;}
                    case 1:{
filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
                            {
                                var input = row.Cells[1].Value.ToString();
                                var list = input?.Split('-').Select(int.Parse).ToList();
                                if (list!.Count == 1)
                                {
                                    return 11 <= list[0] && list[0] <= 50;
```

```
                    }
                    return 11 <= list[0] && list[0] <= 50 || 11 <= list[1] && list[1] <= 50;
                }));
                break;
            }
        case 2:
            {
filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
                {
                    var input = row.Cells[1].Value.ToString();
                    var list = input?.Split('-').Select(int.Parse).ToList();
                    if (list!.Count == 1){
                        return 51 <= list[0] && list[0] <= 200;}
                    return 51 <= list[0] && list[0] <= 200 || 51 <= list[1] && list[1] <= 200;}));break;
            }
        case 3:{
filteredRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row =>
                {
                    var input = row.Cells[1].Value.ToString();
                    var list = input?.Split('-').Select(int.Parse).ToList();
                    if (list!.Count == 1)
                    {
                        return list[0] >= 201;
                    }
                    return list[0] >= 201 || list[1] >= 201;
                }));
                break;
            }  default:
            LoadDataGrid();
            return;
        }
        ConfigurationsDataGridView.Rows.Clear();
        foreach (var row in filteredRowsCollection)
        {
            ConfigurationsDataGridView.Rows.Add(row);
        }}
    private void LoadDataGrid()
    {
        ConfigurationsDataGridView.Rows.Add("SPA (Single Page Application)", "1", "Landing;\nUsually used for product promotion purposes", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 4, параметр Crawl strategy – значення Fastest;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 15, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр
```

Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Low\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Governmental services", "1-50", "Usually used for highlighting results of their job for citizens", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 10, параметр Crawl strategy – значення Normal;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 150, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Critical\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Local online stores", "51-200", "Usually used for selling goods via internet", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 6, параметр Crawl strategy – значення Most complete;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 150, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Medium\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Global online stores", "200+", "Usually used for selling goods via internet", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth –значення 10, параметр Crawl strategy – значення Most complete;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 150, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n

блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - High\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Forums", "201+", "Usually used for discussing different topics", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 4, параметр Crawl strategy – значення Most complete;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 150, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Normal\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Mass media", "201+", "Usually used for highlighting the latest news behind various topics", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 5, параметр Crawl strategy – значення Normal;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 30, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - High\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("File hostings", "201+", "Usually used for sharing files between the users of the internet", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 7, параметр Crawl strategy – значення More complete;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 60, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions: лишається незмінним;\r\n блок Handling Application Errors During Crawl: лишається незмінним;\r\n блок Crawl Project Option Overrides: лишається незмінним;\r\n блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n блок Issues Reported: лишається незмінним;\r\n блок Handling Application Errors During Audit: лишається незмінним;\r\n блок Insertion Point Types: вибрати всі опції;\r\n блок Modifying Parameter Locations: вибрати всі опції;\r\n блок Ignored Insertion Points: лишається незмінним;\r\n блок Frequently Occurring Insertion Points: лишається незмінним;\r\n блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - High\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Automated control systems", "51-200", "Usually used for production systems management", "For Burp Suite:\n блок Crawl Optimization: параметр Maximum link depth – значення 4, параметр Crawl strategy – значення Normal;\r\n блок Crawl Limits: параметр Maximum crawl time – значення 30, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n блок Login Functions:

лишається незмінним;\r\n□ блок Handling Application Errors During Crawl: лишається незмінним;\r\n□ блок Crawl Project Option Overrides: лишається незмінним;\r\n□ блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "□ блок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n□ блок Issues Reported: лишається незмінним;\r\n□ блок Handling Application Errors During Audit: лишається незмінним;\r\n□ блок Insertion Point Types: вибрати всі опції;\r\n□ блок Modifying Parameter Locations: вибрати всі опції;\r\n□ блок Ignored Insertion Points: лишається незмінним;\r\n□ блок Frequently Occurring Insertion Points: лишається незмінним;\r\n□ блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Critical\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Knowledge bases", "201+", "Usually used for highlighting results of their job for citizens", "For Burp Suite:\n□ блок Crawl Optimization: параметр Maximum link depth – значення 5, параметр Crawl strategy – значення More complete;\r\n□ блок Crawl Limits: параметр Maximum crawl time – значення 60, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n□ блок Login Functions: лишається незмінним;\r\n□ блок Handling Application Errors During Crawl: лишається незмінним;\r\n□ блок Crawl Project Option Overrides: лишається незмінним;\r\n□ блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n□ блок Issues Reported: лишається незмінним;\r\n□ блок Handling Application Errors During Audit: лишається незмінним;\r\n□ блок Insertion Point Types: вибрати всі опції;\r\n□ блок Modifying Parameter Locations: вибрати всі опції;\r\n□ блок Ignored Insertion Points: лишається незмінним;\r\n□ блок Frequently Occurring Insertion Points: лишається незмінним;\r\n□ блок Misc Insertion Point Options: лишається незмінним;\r\n\r\nДля Acunetix:\nПоле Business Criticality - Medium\nПоле Default Scan Profile - Full Scan\n");

ConfigurationsDataGridView.Rows.Add("Neural networks", "1-50", "Usually used for highlighting results of their job for citizens", "For Burp Suite:\n□ блок Crawl Optimization: параметр Maximum link depth – значення 4, параметр Crawl strategy – значення Normal;\r\n□ блок Crawl Limits: параметр Maximum crawl time – значення 30, параметр Maximum unique locations discovered – значення 1500, параметр Maximum request count – значення лишається пустим;\r\n□ блок Login Functions: лишається незмінним;\r\n□ блок Handling Application Errors During Crawl: лишається незмінним;\r\n□ блок Crawl Project Option Overrides: лишається незмінним;\r\n□ блок Miscellaneous: лишається незмінним.\r\n\r\nFor Acunetix:\r\n\r\nПараметр Limit Crawling to address and sub-directories only - активувати;\r\nПараметр Excluded Paths - не змінювати;", "For Burp Suite:\nблок Audit Optimization: параметр Audit speed – значення незмінне, параметр Audit accuracy – значення Minimize false positives;\r\n□ блок Issues Reported: лишається незмінним;\r\n□ блок Handling Application Errors During Audit: лишається незмінним;\r\n□ блок Insertion Point Types: вибрати всі опції;\r\n□ блок Modifying Parameter Locations: вибрати всі опції;\r\n□ блок Ignored Insertion Points: лишається незмінним;\r\n□ блок Frequently Occurring Insertion Points: лишається незмінним;\r\n□ блок Misc Insertion Point Options: лишається незмінним;\r\n\r\n\r\nДля Acunetix:\nПоле Business Criticality - Normal\nПоле Default Scan Profile - Full Scan\n");}private void SearchButton_Click(object sender, EventArgs e){

LoadDataGrid(); var rowsCollection = ConfigurationsDataGridView.Rows;

var searchedRowsCollection = new List<DataGridViewRow>();

searchedRowsCollection.AddRange(rowsCollection.Cast<DataGridViewRow>().Where(row => row.Cells[0].Value.ToString()!.ToLower().Contains(SearchBarTextBox.Text.ToLower())));

ConfigurationsDataGridView.Rows.Clear();

foreach (var row in searchedRowsCollection)

{ConfigurationsDataGridView.Rows.Add(row);}}