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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«МЕТОДИКА ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ»

Виконав: студент 2-го курсу, групи 1БС-20м
спеціальності 125 – Кібербезпека
(шифр і назва напряму підготовки, спеціальності)
_____ Душко А.О.
(прізвище та ініціали)
Керівник к.т.н., доц. каф. ЗІ
_____ Баришев Ю. В.
(прізвище та ініціали)
Опонент: к.т.н., доц. каф . ОТ
_____ Савицька Л. А.
(прізвище та ініціали)

«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ЗІ
д.т.н., проф.
_____Лужецький В. А.
«____»_____2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний університет
Факультет Інформаційних технологій та комп'ютерної інженерії
Кафедра Захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань 12 Інформаційні технології
Спеціальність 125 Кібербезпека
Освітньо-професійна програма Безпека інформаційних і комунікаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д.т.н., проф.
_____ В.А. Лужецький
_____ _____ 2021 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Душко Аліні Олександрівні

1. Тема роботи: «Методика тестування безпеки веб-застосунків» керівник роботи: Баришев Юрій Володимирович, доц. каф. ЗІ,
затверджені наказом ректора ВНТУ № 277 від 24 вересня 2021 р.
2. Строк подання студентом роботи 17 грудня 2021 р.
3. Вихідні дані до роботи:
– методика повинна відповідати стандартам ISO, NIST, COBIT, OWASP;
– застосування методики при тестуванні веб-застосунків;
– застосування методики під час розробки веб-застосунків;
– підтримка інструментів для тестування безпеки.
4. Зміст розрахунково-пояснювальної записки:
Вступ. Аналіз інформаційних джерел. Формування вимог до методики тестування безпеки. Розробка методики тестування безпеки веб-застосунків. Застосування методики тестування безпеки для веб-застосунків. Економічна частина. Висновки. Перелік використаних джерел. Додатки.
5. Перелік ілюстративного матеріалу.
Результати аналізу стандартів (плакат, А4). Результат формування вимог до методики тестування безпеки веб-застосунків (плакат, А4). Приклад готового чек-ліста (плакат, А4). Приклад готових тест-кейсів (плакат, А4). Алгоритм застосування методики (плакат, А4). Алгоритм проведення перевірки безпеки за відібраним тест-кейсом (плакат, А4). Результати тестування (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанти	Підпис, дата	
		завдання видав	завдання прийняв
1	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
2	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
3	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
4	Баришев Ю. В., к. т. н., доц. каф. ЗІ		
5	Лесько О. Й., к. е. н., проф., зав. каф. ЕПВМ		

7. Дата видачі завдання 9 вересня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання. Вступ	01.09.2021 – 04.09.2021	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2021 – 15.09.2021	
3	Науково-технічне обґрунтування	16.09.2021 – 22.09.2021	
4	Розробка технічного завдання	23.09.2021 – 04.10.2021	
5	Аналіз методик та методів тестування безпеки	05.10.2021 – 08.10.2021	
6	Аналіз та формування вимог до методики	09.10.2021 – 16.10.2021	
7	Розробка методики тестування безпеки веб-застосунків	17.10.2021 – 14.11.2021	
8	Застосування методики тестування безпеки веб-застосунків	15.11.2021 – 17.11.2021	
9	Розробка розділу економічного обґрунтування доцільності розробки	18.11.2021 – 21.11.2021	
10	Аналіз виконання ТЗ, висновки	22.11.2021 – 24.11.2021	
11	Оформлення пояснювальної записки	25.11.2021 – 29.11.2021	
12	Попередній захист та доопрацювання МКР	30.11.2021 – 12.12.2021	
13	Перевірка магістерської роботи на наявність плагіату	13.12.2021 – 14.12.2021	
14	Представлення МКР до захисту, рецензування	15.12.2021 – 20.12.2021	
15	Захист МКР	21.12.2021 – 23.12.2021	

Студент _____ А. О. Душко

Керівник роботи _____ Ю. В. Баришев

АНОТАЦІЯ

УДК 004.056:004:054

Душко А. О. Методика тестування безпеки веб-застосунків. Магістерська кваліфікаційна робота зі спеціальності 125 – кібербезпека, освітня програма – Безпека інформаційних та комунікаційних систем. Вінниця: ВНТУ, 2021. 84 с.

Укр. мовою. Бібліогр.: 38 назв; рис.: 4; табл.: 14.

Магістерська кваліфікаційна робота присвячена розробці методики тестування безпеки веб-застосунків на основі діючих стандартів щодо забезпечення безпеки інформаційно-комунікаційних систем. Під час розробки методики проведено аналіз існуючих методів та методик тестування безпеки, проаналізовано ризики та загрози щодо роботи веб-застосунків, та зібрано ряд вимог, на основі діючих стандартів та результатів аналізу ризиків, яким повинна відповідати розроблювана методика. Даний спосіб тестування безпеки веб-застосунків дозволяє перевірити безпеку не як частину функціональності, а як стан застосунку. Під час розробки було проведено тестування даної методики, результат якого, було проаналізовано та зроблено оцінку отриманих даних відповідно до розроблених для методики очікуваних результатів.

Графічна частина складається з 7 плакатів з демонстрацією проміжних результатів розробки, кінцевого результату та проведених дослідів.

В економічному розділі оцінено доцільність використання даної методики та витрати на її розробку.

Ключові слова: безпека, тестування, веб-застосунок, вимога, чек-ліст, тест-кейс, методика, контроль якості.

ABSTRACT

Dushko A. Methodology of web application security testing. Master's thesis in the specialty 125 - cybersecurity, educational program - Security of information and communication systems. Vinnytsia: VNTU, 2021. 84 p.

In Ukrainian language. Bibliogr .: 38 titles; fig .: 4; tab .: 14.

The master's qualification work is devoted to the development of methods for testing the security of web applications based on current standards for the security of information and communication systems. During the development of the methodology, the analysis of existing methods and techniques of security testing was analyzed, the risks and threats to the operation of web applications were analyzed, and a number of requirements were collected, based on current standards and risk analysis results. This method of testing the security of web applications allows you to test security not as part of the functionality, but as the state of the application. During the development, testing of this method was performed, the result of which was analyzed and evaluated the data obtained in accordance with the expected results developed for the method.

The graphic part consists of 7 posters demonstrating the intermediate results of development, the final result and the experiments.

The economic section evaluates the feasibility of using this technique and the cost of its development.

Keywords: security, testing, web application, requirement, checklist, test case, technique, quality control.

ЗМІСТ

ВСТУП

На сьогоднішній день багато провідних компаній світу переводять свою роботу в онлайн режим. Це також стосується комерційних компаній, які переводять свій бізнес в режим онлайн. Наприклад, якщо колись, для того щоб купити велику кількість товарів потрібно було їхати до магазину, дивитися, обирати, оформляти купу паперів для підтвердження великої покупки, то зараз, це все можна зробити навіть не виходячи із дому [1]. Навіть деякі державні послуги переводять в режим онлайн для зручності та безпечності користування ними людьми, тому постає питання щодо безпечності послуг онлайн.

Відповідь на це питання є неоднозначною, адже з одного боку так, це буде безпечно, якщо розробники подбали про функціональні методи захисту, по типу шифрування чи автентифікації користувача. Тоді це буде перевірка безпеки як частини функціональності застосунку. Зазвичай, багато хто на цьому зупиняється і забуває про те, що існує чимала кількість способів зруйнувати або дістати інформацію, обійшовши функціональні методи захисту. Дана проблема набула такого масштабу, що деякі її аспекти були включені в стандарти щодо кібербезпеки ETSI TS 103 645 V1.1.1 (2019-02) [2].

Звідси постає задача, як саме перевірити безпеку застосунку, але не як частину функціональності. **Актуальним** вирішенням даної проблеми може стати розробка методики тестування безпеки веб-застосунків, яка допоможе перевірити безпеку застосунку як один із його станів.

Наразі не існує однієї єдиної методики тестування безпеки веб-застосунків, яка б перевіряла не тільки безпеку як частину її функціональності. Загалом зараз тестування безпеки в різних джерелах відносять до функціонального та не функціонального видів тестування [3]. Тому й способів як тестувати безпеку, як один із станів застосунку, немає.

Дану методику можна буде застосовувати не тільки після виходу готового продукту, але й під час його створення.

Об’єктом дослідження є процес тестування безпеки веб-застосунків.

Предметом дослідження є методика тестування безпеки веб-застосунків.

Метою магістерської кваліфікаційної роботи є збільшення якості перевірки безпеки веб-застосунків за рахунок створення нової методики тестування безпеки, яка базується на діючих стандартах щодо забезпечення безпеки для інформаційно-комунікаційних систем, результатів дослідження можливих ризиків та загроз, а також розроблених чек-лістів та тест-кейсів.

Для досягнення даної мети, потрібно вирішити ряд задач:

- проаналізувати відомі види та методики процесу тестування, стандарти NIST, ISO, COBIT, OWASP;
- проаналізувати можливі ризики та загрози;
- проаналізувати існуючі інструменти тестування безпеки;
- сформувати вимоги, розробити чек-листи та тест-кейси для тестування безпеки веб-застосунків;
- сформувати очікувані результати методики;
- виконати перевірку коректності застосування методики.

Наукова новизна вперше запропоновано метод тестування безпеки веб-застосунків, який на відміну від відомих розглядає стан безпеки веб-застосунку як окремий показник якості, а не як частину функціональності, що дозволяє приймати більшу множину рішень за результатами тестування.

Практична цінність: дану методику можна буде застосовувати не тільки після отримання фінальної версії застосунку, але й під час самої його розробки, що на відміну від відомих методик та методів тестування, є удосконаленням існуючих та буде давати більш якісніші результати тестування, а виявлені проблеми під час розробки можна буде усунути з меншою шкодою аніж під час перевірки фінальної версії.

За результатами розробки були **опубліковані** тези, що доповідалися на конференціях XLIX науково-технічної конференції підрозділів ВНТУ [4] та Молодь в науці: дослідження, проблеми, перспективи (МН-2021) [5].

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1.1 Аналіз роботи веб-застосунків

Мережа Інтернет надає можливість здійснювати різні речі не виходячи з дому. Замовити їжу, придбати речі, проконсультуватися з лікарем чи оплатити комунальні послуги – усе це сьогодні можна зробити за допомогою будь-якого пристрою, який має підключення до мережі Інтернет [1]. Вони можуть використовувати або веб-застосунок або ж веб-сайти. Але між ними є певна відмінність.

Веб-сайт – це група глобально доступних, пов'язаних між собою веб-сторінок, які мають одне доменне ім'я [6, 7]. Вони можу бути розроблені однією людиною, бізнесом чи компанією. Підтримка таких веб-сайтів також забезпечується певною командою людей або лише однією особою. Веб-сайт може бути використаний для різноманітних цілей, таких як: купівля харчових чи не харчових товарів, оплата рахунків чи замовлення послуг, контролювати дії певної групи користувачів, якщо таке передбачено в політиці компанії тощо.

Веб-сайт може розміщуватися на одному або декількох веб-серверів. Він буде доступний через мережу, наприклад, мережу Інтернет або локальну мережу через попередньо визначену IP-адресу. Основні причини для яких використовується веб-сайт можна назвати такі: це ефективний спосіб продемонструвати свої продукти та послуги, допомога в брендингу та просуванню бізнесу, дозволяє підтримувати велику кількість клієнтів одночасно, можна легко найти інформацію за допомогою пошукових систем тощо.

Але веб-сайт має низку недоліків, які не завжди беруть до уваги не тільки користувачі, але й розробники [6, 7, 8]. Веб-сайт може припинити працювати через велику кількість користувачів, якщо цього не передбачили розробники. Форма підтримки та зв'язку, яка опублікована на веб-сайті може стати гарним інструментів для хакерів, або для тих користувачів, які можуть надіслати багато небажаних листів. Інформацію на будь-якому веб-сайті може бути ненадійною та неправдивою, якщо вона немає підкріплених доказів або ж оновлюється

нерегулярно. Для інформаційних сайтів автентифікація може бути необов’язковою, але сайт може містити активні вікна чи кнопки, які можуть бути використані для отримання чи сфальсикування даних користувачів або компаній. Веб-сайт це є кінцевий продукт, який потребує оновлення відповідно до розвитку нового апаратного забезпечення користувачів або розробників, але не всі мають можливість оновити сайт, якщо щось було змінено серед пристроїв.

Веб-застосунок – це програмний застосунок або програмне забезпечення, яке доступне через будь-який браузер [8, 9]. Його інтерфейс зазвичай створюється за допомогою різних мов програмування та формування розмітки або стилів, таких як: HTML, CSS, Javascript тощо, що підтримуються багатьма браузерами. Веб-застосунок отримав свою популярність після появи руху SaaS або «Програмне забезпечення як послуга» [10], що проголошує модель надання споживачам ряд програм, при якій потрібно розробити веб-застосунок, розмістити її та керувати нею лише за допомогою мережі Інтернет, що спрощує її використання споживачами.

Веб-застосунок більш популярний аніж веб-сайт чи десктоп-застосунок. І на це є декілька причин [11]. Однією з головних причин є те що, їх легше підтримувати, так як вони використовують один і той самий код у всьому застосунку, тому це усуває багато проблем із сумісністю інших застосунків. Веб-застосунок може використовувати для роботи будь-яку платформу чи операційну систему, так які усі вони підтримують сучасні браузери. Його оновлення може бути здійснено в будь-який час та будь-який момент, при цьому не потрібно сповіщати користувачі про це. Веб-застосунок це економічний варіант для компаній, бо можна купувати не усю ліцензію на використання та розробку, а оплачувати її поступово. Їх не потрібно скачувати чи встановлювати, адже можна використовувати лише браузер. Його можна легко протестувати за допомогою автономних тестів, а також веб-застосунок має велику масштабованість, що є важливою властивістю для сучасного світу.

Веб-застосунок також має низку недоліків, серед яких можна виділити наступні [11, 12]. Не завжди є гарантія того, що застосунок є безпечним для

користувачів, наприклад він може бути вразливим для несанкціонованого доступу. Веб-застосунок може бути створений для спеціальної операційної системи, тому його важко знайти в магазині застосунків іншої операційної системи. Він має обмежений доступ до функцій пристрою, наприклад завантаження файлів до веб-застосунку чи щось інше, якщо це звісно не передбачено розробниками.

Безпека для веб-сайту та для веб-застосунку є важливою складовою для забезпечення нормального використання їх користувачами [13]. Найбільш розповсюдженими загрозами для веб-сайту виділяють: міжсайтовий скриптинг (XSS), SQL ін’єкції, підробка міжсайтових запитів (CSRF), відмова в обслуговуванні (DoS/DDoS), обхід керуючих каталогів (Directory Traversal), ін’єкція керуючих команд (Command Injection) та інші. Здійснення таких атак несе за собою зміну, знищення, часткове або повне видалення даних, або ж їх викрадення. Так, наразі існує багато методів запобігти цим атакам, але майже усі вони є функціональними, і спрямовані на конкретні види атак.

Найбільшими загрозами безпеки для веб-застосунків можна назвати ті самі загрози як і для веб-сайтів. Але можна виділити такі специфічні загрози як: неточне та недостатнє введення журналу успіху, яке може привести до незнання джерела проблеми, неправильна конфігурація застосунку, яка впливає на використання конфіденційної інформації, використання при розробці компонентів, які можуть стати гарним джерелом для початку роботи зловмисника. Зловмиснику не важливо буде це сайт чи застосунок, у будь-якому випадку, якщо він знатиме як це зробити, він обов’язково це зробить.

Веб-застосунок є більш специфічним, аніж веб-сайт [6]. Веб-сайт стає доступним для усіх хто має його адресу, і націлений на загальні потреби користувачів. Веб-застосунок в основному створюється для конкретних цілей: спілкування всередині компаній, контроль процесів, збереження чи обмін файлами та будь-якою іншою інформацією.

Проаналізувавши переваги та недоліки, а також загрози для веб-сайту та веб-застосунку, можна сказати, що вони є дуже схожими між собою. Вони мають

майже однакові недоліки та переваги, а загрози для обох можуть стати руйнівними. Веб-сайт та веб-застосунок в основному можуть бути написані на специфічній платформі, яка передбачає різні методи захисту. Але веб-застосунок – це більш специфічна форма для інформації в Інтернеті, і саме для неї не завжди підходять ті методи захисту, які є для веб-сайту.

В обох випадках захист впроваджується опираючись на функціональні характеристики. Наразі, багато компаній прагнуть перевести свій бізнес або процеси всередині компаній у веб-застосунок. І тому безпека для нього є важливіша, аніж для веб-сайту. Адже веб-сайт це більше про загальний доступ усіх людей, які мають на меті купити, продати, чи щось оформити. Вирішенням цієї проблеми може стати розробка методики перевірки безпеки веб-застосунку, яка буде націлена на перевірку стану безпеки в цілому. А впровадження даної методики допоможе підвищити безпеку не тільки з функціонального боку, але й в цілому.

1.2 Аналіз видів тестування

Тестування будь-якого програмного забезпечення може бути різним. Існує багато видів тестування, які переплітаються між собою, але мають діже мілкі, але все ж відмінності. Для того щоб краще зрозуміти, що таке тестування безпеки потрібно проаналізувати які взагалі існують види тестування.

Для початку тестування поділяють на динамічне та статичне тестування [3, 14]. Динамічним тестування є тестування, яке відбувається із запуском коду на виконання. Запускатися код може як повністю, так і деякі його модулі чи функції. Статичне тестування – це тестування без запуску коду. В різних джерелах можуть назвати статичним тестування – збір та аналіз вимог до початку розробки програмного забезпечення.

За доступом до коду та архітектурою застосунку виділяють: метод білої скрині, метод чорної скрині та метод сірої скрині [3, 14]. Метод білої скрині (white box testing) передбачає доступ до коду. Тобто у тестувальника є доступ до внутрішньої структури коду. Він може запустити на виконання код, та одночасно

дивитися у нього. Які данні він буде повертати під час його роботи. Для цього тестувальнику потрібно розуміти не тільки, як він працює, а як було побудовано його структуру, а також мати базові навички з програмування.

Метод чорної скрині (black box testing) передбачає тестування без доступу до коду. В такому випадку у тестувальника немає доступу до внутрішньої структури коду, або в нього немає достатньо знань для розуміння написаного, або ж він свідомо до нього не звертається у процесі тестування.

Метод сірої скрині (grey box testing) це щось середнє між методом білого та чорного ящиків. Тобто у тестувальника є доступ до частини архітектури коду, а до частини немає. Іноді в різних джерелах, даний вид тестування взагалі пропускають, так, як не вважають його чимось важливим для розуміння. Але в основному при використанні методу білої та чорної скрині, все одно виходить, що тестування відбувається за методом сірої скрині.

За ступенем автоматизації виділяють ручне та автоматизоване тестування [3, 14]. Ручне тестування (manual testing) – це тестування в якому тест-кейси безпосередньо виконуються вручну тестувальником без використання допоміжних застосунків для автоматизованого тестування. Автоматизоване тестування (automated testing) це більше про набір технік, підходів та інструментів, які дозволяють виключити людину із процесу тестування. Але такий вид потребує постійного оновлення тестів, так як код має властивість постійно змінюватися.

За рівнем деталізації застосунку або за рівнем тестування виділяють такі види: модульне, інтеграційне, системне [3, 14]. Модульне (unit testing, component testing) тестування направлено на перевірку окремих невеликих частин застосунку, які як правило можна тестувати ізольовано від інших. Зазвичай таке тестування виконує розробник.

Інтеграційне тестування (integration testing) направлено на перевірку взаємодії між декількома частинами застосунку, кожна з яких була перевірена на стадії модульного тестування. Іноді при перевірці окремих модулів все може працювати добре, але при їх взаємодії можуть виникнути проблеми, тому це вид

тестування є важливим. Системне тестування (system testing) перевіряє роботу всього застосунку в цілому, як застосунок буде працювати після перевірки двох попередніх стадій.

За класифікацією ступені важливості тестувальних функцій виділяють димове тестування, тестування критичного шляху та розширене тестування [3, 14]. Димове тестування (smoke testing) перевіряє роботу найголовніших функцій застосунку, відмова яких може привести до руйнування усього застосунку. Тестування критичного шляху (critical path testing) перевіряє функціональності, які є типовими для користувача при його типовій послідовності виконання дій. Розширене тестування (extended testing) направлено на перевірку всіх, заявлених у вимогах функціональності, навіть тих, які мають найнижчий пріоритет до виконання.

Відповідно до класифікації за принципом роботи застосунку виділяють позитивне та негативне тестування [3, 14]. Позитивне тестування (positive testing) досліджує роботу застосунку при нормальних та правильних умовах, тобто так як ніби дії користувача будуть виконуватися строго за інструкцією. Негативне тестування (negative testing) направлено на перевірку роботи застосунку при невластивих йому умовах. Тобто коли для застосунку будуть виконані некоректні, неправильні, помилкові операції, або ж використовуватимуться дані, які потенційно будуть приводити до помилок.

За класифікацією за метою та задачам виділяють негативне, позитивне, функціональне, нефункціональне, повторне, регресивне, тестування доступності, тестування безпеки та інші. Ця класифікація включає в себе в різних джерелах від 10 до 30 видів тестування [3, 16, 17]. Усі вони переплітаються між собою, а також можуть відноситися до інших класифікацій.

Функціональне тестування (functional testing) перевіряє коректність роботи функцій застосунку. Тобто перевіряється які функції застосунку працюють коректно, а які ні. Часто функціональне тестування асоціюють з тестування по методу чорного ящика. В даному випадку можуть перевірятися різні функції: від відкриття файлу до його завантаження, від функцій шифрування та

розшифрування до перевірки усього алгоритму автентифікації тощо. Найчастіше тестування безпеки відносять до даного виду тестування, так як багато хто вважає, що перевірка шифрування даних чи алгоритму роботи автентифікації це вже і є перевірка безпеки, хоча це не так.

Нефункціональне тестування (non-functional testing) – вид тестування, що направлений на перевірку нефункціональних властивостей застосунку, тобто коректність реалізації нефункціональних її частин, таких як зручність використання, сумісність з іншими застосунками, безпека, масштабованість тощо. Безпека в даному виді перевіряється як стан застосунку, але майже в усіх джерелах лише зазначено про це, але не наведено як саме проводити тестування безпеки як нефункціональну частину застосунку.

Повторне тестування (re-testing) направлено на виконання тих тест-кейсів, виконання яких раніше привело до виявлення дефекту в роботі застосунку. Регресивне тестування (regression testing) направлено на перевірку того фактору, що в раніше працюючій функціональності не з’явилися нові помилки, що були викликані зміною в коді або середовищі його функціонування. Тестування доступності (accessibility testing) перевіряє наскільки застосунок є придатним для використання людьми з обмеженими можливостями.

Тестування безпеки (security testing) направлене на перевірку здатності застосунку протидіяти спробам зловмисників отримати доступ до персональних даних користувачів чи функцій, які можуть привести до руйнівних наслідків [3, 15]. Ціллю такого тестування є виявлення всіх можливих проблем та слабких місць в застосунку, усунення яких дозволить забезпечити безпечність користування застосунком, а також унеможливить розробникам понести великі втрати на усунення можливих проблем. Даний вид тестування також має свою класифікацію. Відповідно до методології тестування безпеки з відкритим вихідним кодом (OSST) виділяють сім основних видів тестування безпеки [15, 17]: сканування вразливостей, сканування безпеки, тестування на проникнення, оцінювання ризиків, проведення аудиту безпеки, етичний злам застосунку, оцінка стану.

Сканування вразливостей (vulnerability scanning) робиться за допомогою інструментів автоматизованого тестування, які сканують усю систему або ж застосунок на наявність відомих вразливостей. Сканування безпеки (security scanning) включає в себе виявлення слабких місць системи чи застосунку, а потім пропонує вирішення для зниження ризику слабких місць. Таке сканування може виконуватися як для ручного так і для автоматизованого тестування.

Тестування на проникнення (penetration testing) – це вид тестування, який моделює атаку зловмисника на систему [15, 18]. Даний вид тестування включає в себе аналіз конкретної системи чи застосунку на предмет потенційних слабких місць при спробі зламу ззовні. Оцінювання ризиків (risk assessment) включає в себе аналіз ризиків безпеці всередині компанії. Далі ризики класифікуються по пріоритетністю та на основі аналізу рекомендує способи вирішення чи мінімізування цих ризиків.

Проведення аудиту безпеки (security auditing) забезпечує внутрішню перевірку застосунків чи систем, що використовуються під розробки, на предмет вразливостей. Етичний злам застосунку (ethical hacking) – це тестування, що передбачає злам систем програмного забезпечення компанії розробника застосунку. Ціль такого тестування виявити недоліки в системі безпеки компанії, які можуть вплинути на їх розробку застосунку. Оцінка стану (posture assessment) об’єднує сканування безпеки, етичний злам застосунку та оцінювання ризиків, щоб показати загальний стан безпеки системи чи застосунку.

Проаналізувавши основні види тестування, можна сказати, що тестування безпеки немає свого конкретного місця в ієрархії видів тестування. Його відносять як до функціонально так і до нефункціонального видів тестування, також відносять в окремий вид [3, 14]. Багато хто вважає що тестування безпеки та тестування на проникнення це одне й те саме, та й навіть різні джерела пишуть по різному. Тому немає взагалі однієї єдиної класифікації тестування, що й затрудняє розуміння куди ж відноситься тестування безпеки та що воно взагалі таке, але при цьому між ними є колосальна різниця. Якщо розглядати тестування на проникнення та тестування безпеки, то перше включає в себе специфічні дії,

які виконуються для моделювання дій зловмисника [18, 19]. Результатом такого тестування стане звіт, який повинен містити в собі низку виявлених ризиків та вразливостей, використаних методів атаки та рекомендації щодо їх усунення. Воно включає в себе не тільки допоміжні автоматизовані засоби тестування, а також дані, персонал, документацію та інші процеси. Тобто результат такого тестування залежить не тільки від того що було використаного під час його проведення, але й від метрик системи, умов налаштування, коректності дій персоналу, сумісності усіх процесів тощо.

В той же час тестування безпеки це комплексний вид тестування, який має ітеративну структуру, яка включає в себе багато комплексних видів тестування, що були описані раніше [17]. Тестування безпеки перевіряє безпеку як стан, так і як функціональність застосунку чи системи. І тому тестування на проникнення не можна вважати повним тестуванням безпеки, воно буде лише однією із його складових. При цьому тестування безпеки потрібно починати на ранніх етапах розробки застосунку чи системи, адже забезпечення безпеки на усіх етапах їх розробки дозволить усунути можливі майбутні збитки.

Через не точну класифікацію видів тестування, багато хто вважає що функціонально перевірити безпеку буде достатньо, але це не так. В той же час перевірка безпеки як стану застосунку зараз майже ніхто не перевіряє, так як через плутанину у видах тестування немає точної методики як саме це зробити. Тому створення методики для тестування безпеки як стану застосунку, яку можна буде застосовувати на усіх етапах його розробки може стати гарним рішенням цієї проблеми.

1.3 Аналіз інструментів для тестування безпеки

Для тестування безпеки виділяють велику кількість інструментів, що допомагають отримати кращі результати. Найпоширенішими інструментами є: Intruder, Owasp Dependency Check, Acunetix, WireShark, W3af.

Intruder – це простий у використанні сканер вразливостей корпоративного або локального рівнів [20]. Він виконує більше ніж 10000 високоякісних

перевірок, що стосуються безпеки, для компанії, системи чи застосунку. Ці перевірки включають в себе: перевірку слабких місць конфігурації, перевірку слабких місць застосунку такі як SQL injections та XSS, а також виявляє відсутні властивості, які можуть стати гарним джерелом для роботи зловмисника. Надаючи інтелектуально розставлені по пріоритетам результати, а також проактивне сканування на наявність останніх загроз, даний інструмент допомагає зберегти час та захищає будь-які системи різного масштабу.

Функції даного інструмента можуть бути різними. Він використовує AWS, Azure та Google Cloud точки приєднання для аналізу, надає якісний звіт зі сканування, має інтеграцію з корпоративними додатками Slack, Microsoft Teams, Jira, Zapier, що дозволяє розширити масштаб сканування, має також функцію інтеграції API з наявними для застосунку CI/CD процесами.

Owasp – це всесвітня некомерційна організація, діяльність якої направлена на підвищення безпеки програмного забезпечення [21]. У даному проекті передбачені низка застосунків для тестування різного роду програмних середовищ на предмет вразливостей, а також протоколів за якими відбувається звітність після його роботи.

Даний інструмент включає в себе Zed Attack Proxy, що є інструментом для тестування на проникнення [22]. Також включає OWASP Dependency Check, що сканує вразливості застосунку та пропонує рішення щодо їх усунення [21]. Та OWASP Web Testing Environment Project що містить збірник засобів для забезпечення безпеки та документацію щодо проведення тестування безпеки.

Acunetix інтуїтивно простий у використанні застосунок, що допомагає компаніям будь-якого масштабу забезпечити безпеку своїх веб-застосунків від дороговартісних витоків даних [23]. Скануючи веб-застосунок він може виявити певний спектр проблем щодо його безпеки, а також допомагає їх усунути.

Він сканує вразливості QWASP, SQL, XSS. Має розширений пошуковий набір процесів, що працює навіть для багатофакторної автентифікації та модулів, які захищені паролем. Може поєднувати декілька типів тестування одночасно, що пришвидшує аналіз та пошук вразливостей, та допомагає знайти ті

вразливості, які інші інструменти не можуть. Може виявити наявний експлойт, що був доданий до веб-застосунку зловмисником. Звітність після аналізу будується на основі стандартів PCI DSS, NIST, HIPPA, ISO 27001 та інших.

WireShark – це інструмент мережевого призначення для сканування мережі та усіх протоколів, що проходять в момент сканування [24]. Він аналізує пакети в режимі реального часу та відображає їх в зручному для користувача форматі. При цьому, кожен пакет даних можна відкрити, та переглянути його детальну інформацію. Має в собі утиліти які дозволяють зручно переглядати виявлену інформацію.

W3af – це веб-застосунок для сканування наявного програмного забезпечення на предмет вразливостей [25]. Його основна ціль модулювати та проводити атаку на систему, а також проводити аудит системи. Він має три основні утиліти: атака, сканування або виявлення, та аудит. При роботі даного застосунку можна виявити вразливості системи. Наприклад, підключений модуль виявлення w3af шукає різні URL адреси для перевірки вразливостей та пересилає їх до модуля аудиту, який потім використовує дані адреси для пошуку вразливостей. Даний застосунок може бути використаний як для перевірки безпеки, тобто її тестування, так і для атаки зловмисником.

Проаналізувавши найпопулярніші інструменти для проведення тестування безпеки, можна сказати що усі вони між собою схожі. Усі ці інструменти мають кінцеву ціль – виявити вразливість в системі чи застосунку. Інформація отримана після їх роботи може стати головним джерелом щодо виявлення та усунення майбутніх вразливостей, що також допомагає розробникам завчасно подбати про безпеку застосунку чи системи.

Так як, вище наведені інструменти полегшують тестування безпеки в цілому, то при розробці методики тестування безпеки веб-застосунку буде передбачено можливість їх використання, що в свою чергу мінімізує фактор того, що при застосуванні розроблюваної методики, буде щось упущено.

1.4 Постановка задачі

Зараз все доступніше стає можливість розвивати свій бізнес за допомогою інтернет-технологій. Для цього створюють веб-сайти та веб-застосунки. Веб-сайти більше підходять для таких цілей, як купівля чи продаж товару, інформаційні публікації чи перегляд статей. Але веб-сайт надає більше функціоналу, аніж веб-сайт. Це по суті програмне забезпечення, яке не потребує встановлення, адже його функціонування відбувається через браузер. Він має багато функціоналу, яке потребує захисту. Але не весь захист може забезпечитися функціоналом.

Для перевірки правильності функціонування застосунку, проводять тестування. До нього також включають тестування безпеки, але тестувальники та розробники не завжди звертаються увагу, що безпеку потрібно перевіряти не як частину функціональності, але й як сатину його стану також. До того ж, є різноманітна кількість джерел, які наводять класифікацію виді тестування, де в одних може бути сказано що тестування безпеки – це функціональне тестування, а в інших навпаки - не функціональне. Звідси постає задача: немає точної класифікації видів тестування, а тому немає методик тестування, які б перевіряли безпеку як стан застосунку.

Розв’язком даної задачі стане розробка нової методики тестування, яка б перевіряла безпеку веб-застосунку як частину одного з його станів. Дана методика повинна мати певні вимоги, які обґрунтовані на діючих стандартах з безпеки та кібербезпеки, що при її застосуванні допоможе якісніше здійснювати перевірку. Дана методика повинна бути розроблена таким чином, щоб її застосування можливо було не тільки під час розробки застосунку, але й під час отримання його фінальної версії продукту. Також повинна бути можливість застосування різних інструментів тестування безпеки, наприклад, вразливостей, що надасть змогу отримати більш детальну інформацію про стан безпеки веб-застосунку.

2 ФОРМУВАННЯ ВИМОГ ДО МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ

2.1 Математичний опис моделі оцінювання процесу розробки

Для узагальнення процесу тестування потрібно описати математичну модель процесу розробки програмного забезпечення та місце тестування у ньому. Для цього було обрано теоретико-множинний підхід, що дозволяє описати множинний підхід, який базується на побудові деякого кортежу змінних, кожна з яких відповідатиме множині існуючих. Дана модель описує процес розробки застосунку як сукупність множин постійних та змінних параметрів даного процесу:

$$D = < R, N, O, A, S, X, Y >, \text{ де} \quad (2.1)$$

D – процес розробки програмного забезпечення;

R – наявні ресурси;

N – оператор моделювання;

O – оператор оцінювання;

A – показники якості;

S – множина стратегій управління;

X – множина вхідних параметрів;

Y – множина вихідних параметрів.

Для розробки програмного забезпечення повинна бути мета розробки. Зазвичай мета розробки програмного забезпечення формується із вимог замовника та цілей, які має задовільнити розроблюваний застосунок. В загальному метою розробки є досягнення наявних показників якості, які мають співвідноситися до цільових показників якості, наприклад, - bility вимоги. Для компанії яка розробляє власний продукт, метою буде не перевищити наявні ресурси (R), а показники якості (A) мають бути досягнуті по максимуму, наскільки це можливо: $R_g \leq R$, $A \rightarrow \max$. Для аутсорсингових компаній, метою стане реалізація наявних показників якості (A) більше аніж цільових (A_g), з використанням мінімальних ресурсів (R): $R \rightarrow \min$, $A \geq A_g$.

Ресурсами R даної математичної моделі виступатимуть множина апаратного та програмного забезпечення, що використовується в ході розробки застосунку. Також у дану множину входить множина використовуваних мов програмування (*Hard*) та середовищ розробки (*Soft*) в процесі створення застосунку: R = {*Hard, Soft, IDE, Lang* …}. Ресурсами будуть множина різних засобів способів, які використовуються у процесі розробки програмного забезпечення.

Оператором моделювання N виступає сукупність функції або програм, що розробляються. Дана величина є відображенням декартового добутку множини вхідних параметрів X та стратегій управління S:

$$N = X \times S = \{(x;\ s) | x \in X \cap s \in S\}, \text{ де} \quad (2.2)$$

N – множина функцій та програм, що розробляються.

Оператор моделювання встановлює залежність між тим які є критерії для розробки програмного забезпечення та тим, як саме буде побудований процес його розробки.

Оператором оцінювання O є показник ефективності, який описує залежність між вихідними параметрами Y та показниками якості A. Даний показник описує, наскільки розроблювана програма реалізована до встановлених вимог, що визначаються критеріями якості.

Показником якості даного опису виступає множина параметрів A, яка є сукупністю розроблених та встановлених перед початком розробки застосунку критеріїв якості. Вони повинні бути досягнуті після завершення розробки програмного застосунку. Такими критеріями якості можуть бути: зручність у використанні, масштабованість, кросплатформеність, можливість подальшого влаштування нового функціоналу тощо.

Множиною стратегій управління даної математичної моделі виступає сукупність параметрів S, які є відображенням різних підходів щодо розробки програмного забезпечення. Такою величиною можуть бути: методологія розробки програмного забезпечення, архітектура програмного забезпечення та шаблони проектування застосунку. Дотримання даних величин є запорукою

того, як саме будуть налаштовані процеси всередині команди під час розробки програмного забезпечення, як саме будуть налаштовані процеси розробки різного функціоналу, та як саме вони будуть між собою зв’язані.

Множиною вхідних параметрів X є сукупність критеріїв щодо тих параметрів, які може використовувати розроблюваний програмний засіб. Такими параметрами можуть бути: обмеження на використовувані пам'ять, час завантаження, час виконання певних операцій, апаратна сумісність тощо.

Множиною вихідних параметрів Y виступає сукупність створених функцій, які за допомогою своєї взаємодії представляють готовий програмний засіб.

Дана математична модель описує загальний підхід до процесу розробки програмного забезпечення. Тобто є якась мета розробки, певні вхідні параметри, та ресурси, за ними розробляються критерії якості та обирається стратегія управління, тобто розробки, а на виході буде отримано якийсь застосунок, що повинен відповідати встановленим вимогам щодо його розробки. Наразі найбільший вплив у даній моделі має саме мета, а не критерії якості.

Більшість застосунків розробляються задля досягнення мети, а про критерії якості часто забувають. Звісно є якась мета, яку потрібно досягти, і якщо після розробки програмного забезпечення дана мета була досягнена це є гарний показником, але розробляти щось задля того, щоб тільки досягнути цю мету є неправильним рішенням. Ще одним мінусом при розробці застосунків виступає те, що при обранні певної стратегії розробки не звертають увагу на ті самі критерії, а особливо безпеки. Обрання стратегії управління чи архітектури створюваного програмного забезпечення відбувається з метою досягнення кінцевого результату, який би задовільнив замовника та користувачів, але при цьому не завжди дотримуються усіх критеріїв якості та вимог.

Зазвичай розробка програмного забезпечення відбувається третіми сторонами, тобто є замовник та є розробники, які наймають у свою команду якусь третю сторону, щоб якнайшвидше досягнути свою мету та отримати прибуток. Але використання великої кількості людей, дуже часто впливає на

якість самого продукту, адже обирається та методологія розробки, яка допоможе досягнути бажаного результату за короткий термін. В такому випадку забезпечення безпеки застосунку відходить на другий план.

Якщо при розробці програмного забезпечення було упущено певні критерії якості, а розробка велася тільки заради мети, а не заради якісного результату, то і тестування буде неякісним, як і оцінка його якості. Існуючі методики тестування намагаються задовільнити мету розробки, тобто перевірити його функціонал, але як саме він був зроблений, які заходи безпеки та критерії якості при цьому були використані та чи були вони дотримані не перевіряється. Тому розробка методики тестування безпеки як стану веб-застосунку буде гарним рішенням для проведення перевірки критеріїв якості безпеки: чи були вони досягнуті та чи відповідають вони встановленим вимогам. А наведена модель процесу розробки програмного забезпечення дозволить врахувати усі критерії при оцінюванні якості та безпечності застосунку.

2.2 Аналіз етапів SDLC для веб-застосунків

Розробка програмного забезпечення лише діяльність пов’язана з написанням коду, це й про безліч аспектів, які потрібно врахувати: вимоги, задачі, мета, як зробити, що саме і коли роботи. Для того, щоб розробка застосунків велася ефективно потрібно структурувати роботу. Для цього існує SDLC – життєвий цикл розробки програмного забезпечення [26, 28]. Він демонструє весь етап розробки застосунку: від зародження ідеї того, що потрібно створити до припинення підтримки даного програмного забезпечення розробниками. Зазвичай виділяють 8 основних етапів. Залежно від команди розробників, життєвий цикл розробки програмного забезпечення може мати і більше етапів [3].

Етап 1 – Ідея. На даному етапі зароджується ідея самого продукту. Даний процес може бути більш формальним, наприклад, брейншторм в компанії, або ж неформальним – розмова з друзями у кафе. Кінцевою метою даного етапу є виділити із сотень ідей лише одну, ту єдину, яка буде реалізована та

задовольнятиме вимоги замовника. Особливістю даного етапу є те що, потрібно проаналізувати не тільки потреби користувачів, але й наявний ринок послуг: чи немає подібних застосунків, наскільки він буде успішним тощо. Для процесу створення веб-застосунків даний етап є одним із важливих, осільки не тільки має бути по-своєму унікальна ідея, але й вона має бути безпечною, тому потрібно знайти рішення, яке б задовольняло потреби користувачів, та при цьому було безпечним відносно використання даних та функціоналу.

Етап 2 – Планування та визначення вимог. Без якісних вимог – немає якісного продукту. Даний етап надає більше деталей до самої ідеї. На даному етапі відбувається створення вимог щодо якості, проводиться аналіз ризиків та загроз, створюються плани валідації та верифікації, визначаються критерії приймання програмного забезпечення, визначається команда створення застосунку та їх ролі, обирається методологія розробки, стек технологій та здійснюється оцінка часу розробки застосунку. Це найважливіший етап життєвого циклу. Для веб-застосунків особливістю даного етапу є також створення карти веб-застосунку, обрання структури контенту веб-застосунку, обрання стеку технологій, щоб можна було швидко замінити чи змінити функціонал та, особливо, методів захисту.

Етап 3 – Дизайн та моделювання. На даному етапі створюється «скелет» продукту на основі відібраних вимог та визначається специфікація по дизайну, тобто описом того що і як повинно розроблятися з технічної точки зору. Саме тут визначаються функціональні методи захисту веб-застосунку. Особливістю даного етапу для веб-застосунків є те, що на передній план тут виступають саме дизайнери контенту та інтерфейсу (UI/UX). Вони здійснюють попередню розробку дизайну веб-застосунку: розміщення кнопок, полів, картинок, меню, інформаційних панель, реклам (якщо такі є), визначають також шрифти та стилі веб-застосунку. Розміщення контенту та функціональних особливостей веб-застосунку є важливішим, аніж для іншого програмного забезпечення, так як, впровадити, наприклад, шкідливий код чи фішингову рекламу на веб-застосунок

набагато легше ніж для звичайного застосунку, а тому безпека на такі випадки має бути вищою.

Етап 4 – Створення контенту. Даний етап життєвого циклу присутній лише для веб-застосунків та веб-сайтів [27]. Створення каналу зв'язку з користувачами через інтерфейс – основна ціль даного етапу. Написання контенту – створення та подання інформації у зручному вигляді для користувача. На даному етапі також остаточно визначаються із поданням та структурою веб-застосунку. Остаточно визначаються розташування кнопок, полів, меню, відбувається оновлення інформації. Найголовніше для веб-застосунку це оновлення інформації та розташування потрібної, так як користувачі в першу чергу звертають увагу на оформлення та наповнення веб-застосунку, на його зручність у використанні та доцільність.

Етап 5 – Написання коду. Розробники отримують вимоги та створюють застосунок який вимагається. Для веб-застосунків розробники тут діляться на два типи back-end та front-end. Перші пишуть логіку із серверної сторони, того як інформація та дії користувача мають оброблятися застосунком, тобто його функціонал. Інші описуються логіку поведінки користувача, та яка інформація та дії користувача мають йому відображатися, тобто відображення реакції застосунку на дії користувача. Даний етап дуже сильно залежить від етапу створення вимог, оскільки чим гірше оформлені вимоги – тим більше часу забере розробка і більше буде дефектів в розробці застосунку. Тут усі кнопки та поля трансформуються на веб-сторінку в інтерактивні елементи. Результат даного етапу є готовий продукт, який надалі має бути протестовано.

Етап 6 – Тестування. Тестувальники на основі вимог аналізують роботу застосунку. Тобто вони перевіряють відповідність готового продукту до описаних раніше вимог. При невідповідності створюється відповідний документ який називається баг-репорт. Під час розробки веб-застосунків даний етап є дуже важливим. Так як, веб-застосунок це те, що доступне через браузер та мережу інтернет, то і користувачів у нього буде багато. Залежно від функціоналу веб-застосунку, одночасно його можуть використовувати тисячі користувачів. А

тому на даному етапі важливо перевірити стрес тестування, навантажувальне тестування, користувацьке тестування, сумісність та продуктивність веб-застосунку [27]. Оскільки веб-застосунок можна використовувати на різних браузерах, то тут важливо також перевірити як застосунок буде працювати на різних пристроях та браузерах. Важливою особливістю є перевірка доступності інформації, адже при неправильній конфігурації коду та самого функціоналу, може скластися така ситуація, коли особисті дані користувача або ж код застосунку потраплять у відкритий доступ.

Етап 7 – Розгортання та підтримка. Після того, як застосунок було протестовано відбувається його інтеграція у середовище використання, налаштування встановлення та запуск. Для веб-застосунків після тестування продукту відбувається розгортання готового продукту на веб-сервері, використовуючи при цьому протокол передачі файлів та тексту. Ще однією особливістю даного етапу є те, що після розгортання застосунку на веб-сервері, розробники ще можуть залишатися на декілька місяців а той років для підтримки продукту. До того ж, веб-застосунку потрібне регулярне оновлення та підтримка актуальної інформації. Може бути таке, що сервер відмовить в обслуговуванні, а отже, розробники мають забезпечити роботу веб-застосунку таким чином, щоб при будь-який збоях дані користувача не були втрачені або доступними усім.

Етап 8 – Ліквідація та закриття. Даний етап є останнім етапом життєвого циклу розробки програмного забезпечення. Для нього відбувається вивід з експлуатації застосунка, його заміна на сучасні аналоги тощо. Для веб-застосунків даний етап дуже рідко присутній. Так як веб-застосунок можна постійно оновлювати відповідно до новіших технологій, контент можна змінювати, а функціонал додавати, і це все без завантаження та оновлення на стороні користувачів, то й ліквідація відбувається рідко. Тим паче, що для веб-застосунків більше притаманно постійно оновлюватися, але майже ніколи не закриватися.

Кожна фаза життєвого циклу веб-розробки важлива, незалежно від того, наскільки маленьким чи великим є проект. Хоча написання коду є дуже

важливим аспектом будь-якого проекту розробки програмного забезпечення, також важливо не нехтувати іншими етапами веб-розробки, такими як дизайн, створення вмісту, контрольні списки безпеки або тести команди забезпечення якості.

Всі етапи процесу веб-розробки однаково важливі. Якщо на будь-якому з етапів, будь то кодування чи обслуговування, не вистачає зусиль, це впливає на веб-застосунок. Але перші етапи, як правило, вважаються вирішальними, оскільки вони передбачають розв’язання стратегічних задач, які визначають напрям і тривалість подальших кроків.

2.3 Критерії оцінки стану безпеки веб-застосунків

Кожен етап життєвого циклу розробки програмного забезпечення має свої особливості щодо розробки веб-застосунку [26, 27]. Деякі етапи на шляху створення веб-застосунку можуть бути відсутніми, а деякі навпаки – притаманні лише для розробки веб-застосунків. Після аналізу кожного етапу життєвого циклу розробки веб-застосунку було виділено низку критеріїв, яким мають відповідати вимоги методики тестування безпеки для веб-застосунків.

Головним критерієм після проходження даного етапу стане формування ідеї розробки веб-застосунку, його цілі та призначення. На даному етапі може бути створений певний mock-up застосунку, тобто візуальна схема розташування елементів застосунку [26]. Якщо немає точної цілі застосунку, то при зборі вимог та їх програмній реалізації може бути розходження в тому, що повинно бути і що вже є. При таких розходженням часто забувають про безпеку, а інколи її навіть не враховують на даному етапі. Якщо немає чіткої цілі застосунку, що і для чого розробляється, то і розробка буде вестися неправильно та хаотично, що й впливає на безпеку застосунку.

На другому етапі життєвого циклу розробки веб-застосунків головним критерієм буде якість зібраних вимог. Для того, щоб розробка застосунку велася чітко із вимогами та реалізація всіх функцій була зроблена безпечно, вимоги мають відповідати своїм критеріям, а саме: вони не повинні бути двозначними,

мають бути повними, мають бути коректно сформовані, повинна бути можливість реалізувати кожну відібрану вимогу, вони мають забезпечувати необхідність їх реалізації, кожна вимога повинна мати свій пріоритет реалізації, а також мати властивість щодо її тестування. За такими критеріями мають відбиратися вимоги щодо розробки веб-застосунку. Особлива увага має приділятися вимогам безпеки, оскільки відсутність однієї або декількох властивостей вимог можуть в майбутньому порушити безпеку всього застосунку. Також при визначенні на даному етапі команди розробників мають бути визначені відповідні доступи до різного роду інформації, тобто як кожен член команди співпрацює з інформацією, яка йому надається. Обраний стек технологій повинен мати свої властивості: він має бути безпечним щодо використання.

На третьому етапі створюється «скелет» веб-застосунку, прописується його структура та визначається технічна специфікація. Саме на даному етапі визначаються критерії реалізації засобів забезпечення безпеки веб-застосунку, а найголовніше попереднє розміщення інформації та керуючих елементів. Головним критерієм для даного етапу є те, що керуючі елементи повинні мати свою функціональність та правильне розташування, що, в першу чергу, впливає на безпеку веб-застосунку. Сформований попередній дизайн та «скелет» застосунку допоможе доповнити вимоги щодо забезпечення безпеки стану веб-застосунку.

На етапі створення контенту, який притаманний лише для веб-застосунків, важливим критерієм щодо формування вимог стане саме доцільність, повнота та актуальність контенту. Тобто, яка інформаціє є актуальною, де її розміщувати, до якої інформації має доступ користувач, яку він може використовувати та розміщувати. За усіма цими критеріями можна перевірити стан безпеки веб-застосунку, а його забезпечення дозволить в першу чергу отримати правильне та безпечне функціонування веб-застосунку.

Під час написання коду веб-застосунку, він буде ділитися на написання коду для back-end та front-end. Головними критеріями даного етапу стануть

правильна реалізація коду. Немає бути написано двозначних функцій, мають для цього бути створені правила написання коду. Для частини front-end даний етап є також важливим. Якщо back реалізує технічне завдання клієнта, технічне та функціональне забезпечення безпеки, то для другого головним критерієм стане реалізація та взаємодія з клієнтом таким чином, щоб конфіденційна інформація не була йому показана та доступна до використання, наприклад, при отриманні якоїсь помилки не повинно відображатись частина коду, а має бути відповідне інформаційне повідомлення. Головним результатом є готовий продукт, який повинен бути реалізований відповідно до створених вимог. Даний етап є найбільшим серед усього циклу розробки веб-застосунку, а тому вимоги мають забезпечувати повну перевірку функціональності та не функціональності застосунку, особливо щодо реалізації безпеки.

Тестування – етап, який має свій життєвий цикл. Він має свої окремі вимоги щодо його проведення та реалізації. Зазвичай тестувальники перевіряють відповідність реалізованого коду щодо встановлених для них вимог. Зазвичай тестувальники перевіряються дизайн веб-застосунків та його функціональну реалізацію, але про такі вимоги як навантаження застосунку чи його продуктивність забувають, а це в першу чергу впливає на безпеку застосунку в цілому. Тому головним критеріям для даного етапу стане перевірка функціональних та нефункціональних вимог, перевірка поведінки застосунку на непередбачувані обставини та зміну середовища використання. Сформовані вимоги для даного етапу повинні бути такими, щоб тестування стану безпеки було проведено якісно та без двосмислових тестувань, щоб вимоги задовольняли перевірку результатів отриманих після проходження кожного етапу життєвого циклу розробки веб-застосунку.

Критеріями щодо формування вимог для тестування стану веб-застосунку для етапу розгортання та підтримки веб-застосунку стануть критерії постійного оновлення інформації та безперервна підтримка роботи веб-застосунку. Вимоги повинні бути сформовані таким чином, щоб перевірка оновлення інформації, підтримка веб-застосунку та, при необхідності, негайне усунення дефектів в його

роботі були забезпечені та проведені якісно, а тому мають бути сформовані вимоги щодо перевірки середовища функціонування веб-застосунку та перевірки забезпечення та реалізації усіх критеріїв приймання програмного забезпечення.

На останньому етапі життєвого циклу розробки веб-застосунків мають бути сформовані такі вимоги, які б забезпечували перевірку роботи веб-застосунку та його суміжних інструментів в ситуації ліквідації та припинення повного функціонування. Хоча даний етап нечасто притаманний при розробці веб-застосунків, але вимоги для нього повинні бути сформовані таким чином, щоб відбувалась перевірка інформації на предмет витоку, пошкодження, видалення чи підміни під час ситуацій, коли застосунок може перестати працювати повністю або частково.

Таким чином було висунуто низку критеріїв, яким мають відповідати вимоги методики перевірки стану безпеки веб-застосунку. Головним критеріям для усіх вимог методики стане їх повнота, недвозначність, можливість реалізації, забезпечення повноти перевірок та їх зрозумілість.

2.4 Формування вимог до методики тестування безпеки

Для формування вимог щодо методики перевірки стану безпеки веб-застосунку потрібно проаналізувати низку стандартів, які містять вимоги безпечної розробки застосунків [29]. Формування вимог на основі стандартів дозволить створити їх таким чином, щоб при використанні методики тестування стану безпеки веб-застосунків було здійснену якісну його перевірку, яка б охоплювала аспекти, що визначені та мають перевірятися відповідно до міжнародних стандартів. Неможливо забезпечити безпеку перевіряючи вимогу лише одного стандарту, тому було відібрано низку стандартів для подальшого їх аналізу: ISO/IEC 27001:2013 [30], NIST 800-53 [31], COBIT 5 Enabling Processes [31], OWASP ASVS 4.0.3 [32] та ETSI TS 103 645 V1.1.1 (2019-02) [2].

Останній стандарт описує вимоги щодо забезпечення кібербезпеки інтернету речей. Цей документ об’єднує широко визнану передову практику безпеки для споживчих пристроїв, підключених до Інтернету, у набір положень

високого рівня, орієнтованих на результат. Метою цього документа є підтримка всіх сторін, які беруть участь у розробці та виробництві споживчого Інтернету речей, з рекомендаціями щодо захисту своїх продуктів. Тому вимоги, які прописані у даному стандарті будуть переглянуті та взяті до уваги, але більша частина вимог буде сформована на основі інших вище перечислених стандартів.

Стандарт ISO 27001:2013 описує вимоги до створення, впровадження чи підтримки, або постійне поліпшення стану безпеки в інформаційно-комунікаційній мережі [30]. З даного стандарту було обрано низку вимог, яким має відповідати розроблювана методика перевірки стану безпеки веб-застосунку.

Класифікація інформації у даному стандарті визначається тим що, інформація має певний рівень захисту, який відповідає її значущості в компанії. Так, як розробка веб-застосунку потребує збір різного роду вимог, а команда розробників використовує різну інформацію під час розробки, то дана вимога має обов’язково бути властива розроблюваній методиці.

Команда розробників веб-застосунку взаємодіє під час розробки з великою кількістю людей, а тому інформація, яка циркулює всередині даного процесу, повинна бути розділена по доступам. Тому вимога щодо забезпечення політики безпеки має бути обов’язкова присутня серед інших вимог методики.

Даний стандарт також висуває вимоги щодо контролю доступу до середовища розробки. Веб-застосунок має перманентно мінливу структуру, яку, потрібно постійно підтримувати та оновлювати, а тому можливе часте оновлення команди розробників. Тому обмеження доступу інформації носіям її обробки, гарантування авторизованого доступу користувачам та розробникам, попередження несанкціонованого доступу до системи та середовищ розробки, а також правила знищення доступів після виходу кожного члена команди повинні бути описаними та дотриманими. Для методики забезпечення стану безпеки веб-застосунку, ця вимога є потрібною, так як веб-застосунок потребує постійної роботи та підтримки.

Веб-застосунок – це безперервний потік інформації, а тому вимога даного стандарту щодо забезпечення неперервності інформаційної безпеки має бути

обов’язково дотримана та влаштована всередину системи менеджменту команди розробників.

Осільки методика розробляється з метою перевірки саме стану безпеки, то вимоги даного стандарту щодо впровадження криптографічних методів захисту, забезпечення охоронної зони місця розробки, та вимоги щодо відношень з постачальниками засобів та апаратного забезпечення можна не приймати до уваги. Але не ватро забувати, що при перевірці функціонального боку застосунку, дані вимоги мають бути дотримані. Аналіз інших вимог представлено в ілюстративній частині 1 – Аналіз стандарту ISO 27001:2013.

Стандарт NIST 800-53 описує міри щодо забезпечення безпеки та конфіденційності для всіх систем, крім тих, що відносяться до національної безпеки [31]. Відповідно до даного стандарту було відібрано та проаналізовано низку вимог, які мають забезпечуватися розроблюваною методикою перевірки стану безпеки веб-застосунку.

Вимога щодо сканування вразливостей системи щодо проникнення та їх подальший аналіз визначає правила проведення компанією власного тестування на проникнення з метою перевірки стану безпеки. Осільки методика розробляється з метою перевірки цього стану, а тестування на проникнення є один із етапів перевірки безпеки, то для розроблюваної методики дана вимога повинна бути забезпечена. Адже веб-застосунок така річ, яка доступна усім, хто має його адресу та підключення до інтернету, а тому ризик проникнення всередину системи є.

Вимога щодо структури тестування, яка перевірятиме безпеку розробки архітектури та дизайну є також важливою вимогою щодо методики тестування стану безпеки веб-застосунку. Так як, одним із етапів розробки веб-застосунку є створення архітектури застосунку, для якого визначається його функціонал та взаємодія компонентів, то і проведення перевірки дотримання цієї вимоги повинне бути забезпечено. Дана вимога забезпечує використання структурного, спеціального апаратного чи програмного забезпечення для проведення тестування.

Веб-застосунок постійно використовує велику кількість інформації, веб-сервери її обробляють, а далі вона циркулює всередині застосунку. На етапі створення контенту критерієм оцінки є актуальність, повнота та доцільність самого наповнення веб-застосунку. Для того, щоб краще забезпечувалась безпека, потрібно дотримуватися ще однієї вимоги даного стандарту, а саме здійснювати моніторинг інформаційної системи, тобто використовувати інструменти логування подій та журналів аудиту безпеки.

Вимога щодо статичного тестування коду, безпосередньо відноситься до вимог розроблюваної методики тестування безпеки як стану веб-застосунків. На етапі написання коду, вимоги мають забезпечувати перевірку функціональної та не функціональної складової застосунку, особливо щодо реалізації безпеки. Осільки розроблювана методика направлена на перевірку стану безпеки, а не його функціональності, то вимога щодо статичної перевірки коду, тобто code-review, рефракторинг коду, перевірка дотримання вимог написання функцій та інше є важливою вимогою даної методики.

Даний стандарт містить вимоги щодо тестування та перевірки безпеки при автоматизованому тестуванні. Так як, автоматизоване тестування націлене на програмну перевірку роботи функціоналу, а дана методика розробляється в іншому ключі, то дані вимоги можна упустити при перевірці стану безпеки. Більше проаналізованих вимог наведено в ілюстративній частині 1 – Аналіз стандарту NIST 800-53.

Стандарт COBIT 5 Enabling Processes описує вимоги, які стосуються оптимізації управління інформаційними технологіями, а саме аудитом та безпекою [32]. Усі ці вимоги описують шляхи перевірки налаштування процесу розробки всередині компанії, а тому найбільше відносяться до тих, які перевіряють не функціональні властивості застосунків.

Серед даних вимог було обрано вимогу щодо аналізу ризиків та загроз. Аналіз ризиків та загроз дуже важливий в процесу розробки веб-застосунків, а тому потрібно розробляти корисну інформацію для підтримки рішень щодо мінімізування ризиків, які враховують важливість різних факторів ризиків та

загроз для роботи застосунку. Веб-застосунок є відкритим, має великий функціонал, а ще велику кількість інформації яку він має обробляти та передавати. При аналізі ризиків та загроз веб-застосунку варто звертати особливу увагу на те як вони впливають на його функціонал в цілому та що саме потрібно мінімізувати. Дотримання цієї вимоги на етапі планування та збору вимог повинен видати результат сформованих вимог, які відповідають вище встановленим критеріям.

Вимога щодо підготовки веб-застосунку для тестування є однією із найважливіших вимог. Тестування має окремий життєвий цикл, а тому, щоб успішно пройти усі його стадії має бути сформовано план тестування, описано необхідні середовища, що використовуються під час тестування, як застосунка в цілому, так і його окремих компонентів. Дана вимога при тестування безпеки веб-застосунку як частини його стану є дуже важливою, адже без грамотно спланованого тестування виникає можливість, що в майбутньому непротестована, тобто пропущена, якась складова веб-застосунку принесе збитки розробникам, а безпека самого застосунку може бути знищеною.

Вимога щодо складання плану приймального тестування забезпечує опис усіх ролей та доступів, які результати мають бути на виході, а сам план має бути схвалений відповідними фахівцями з тестування. Дана вимога задовольняє критерії, які були встановлені для етапу збору вимог, дизайну та тестування. Задля забезпечення належного стану безпеки веб-застосунку, план має бути повним, недвозначним та чітко структурованим, щоб при його використанні вміло розділяти та виконувати декомпозицію інформації інформацію для тестування. Більш детальний аналіз та відібрані вимоги наведено в ілюстративній частині 1 – Аналіз стандарту COBIT 5 Enabling Processes.

OWASP (Open Web Application Security Project) – спільнота, документ, проект або вимоги, що стосуються веб-застосунків, де прописуються вимоги щодо забезпечення безпеки веб-застосунків [33]. Для розробки методики було обрано OWASP ASVS 4.0.3, так як в ньому напряму прописані вимоги щодо забезпечення безпеки саме для веб-застосунків [34]. А тому при формуванні

майбутнього чек-ліста найбільше увагу буде звернуто саме на вимоги, що відібрані для даного стандарту.

Однією з найперших відібраних вимог є вимога щодо забезпечення життєвого циклу безпечної розробки програмного забезпечення. Зазвичай сам SDLC беруть лише до уваги, але не дотримуються усіх етапів розробки, а тим паче критеріїв кожного етапу [34]. Таке відношення до SDLC супроводжується втратою певних даних та якості під час розробки застосунку. Тому дотримання життєвого циклу безпечної розробки, який включає в себе процеси моделювання загроз та ризиків для кожної зміни дизайну чи функціоналу, а також ідентифікація загроз різного роду є важливою вимогою для досягнення якісного результату при тестуванні стану безпеки.

Вимога щодо забезпечення безпечної конфігурацією архітектури є однією з найважливіших вимог. Для веб-застосунків дана вимога визначатиме кількість та взаємозв’язок компонентів застосунку. Осільки сам веб-застосунок складається з безлічі схожих між собою компонентів, то на етапах моделювання, дизайну та написання коду, дотримання цієї вимоги забезпечить чітке відокремлення компонентів різних рівнів довіри, за допомогою чітко визначених засобів керування безпекою, правил брандмауера, шлюзів API, хмарних груп сховищ або подібних елементів.

Даний стандарт визначає ще одну вимогу загального дизайну контролю доступу. Усі доступи, інформація мають бути прописані в політиці безпеки компанії. При розробці веб-застосунків, дана інформація не має потрапити до користувачів, або ж навпаки - інформація користувачів до розробників. Тому дотримання цієї вимоги може відбуватися через існування принципу найменших привілеїв – користувачі повинні мати доступ лише до функцій, файлів даних, URL-адрес, контролерів, служб та інших ресурсів, на які вони мають спеціальний дозвіл. Дотримання цієї вимоги забезпечить захист від підробки та підвищення привілеїв доступу до інформації в процесі розробки та використання веб-застосунку.

Щодо вимог реєстрації конфіденційної інформації, то даний стандарт надає цілий список вимог, які повинні бути дотримані при розробці веб-застосунку. Застосунок обробляє велику кількість інформації, і відповідно до критеріїв етапу підтримки та розгортання веб-застосунку, мають бути встановлені вимоги щодо контролю оброблюваної та показаної інформації. Такою вимогою є те що, веб-застосунок не повинен реєструвати облікові дані користувачів чи деталі платежу, а маркери сеансу користувача, задля подальшого аналізу їх системою та використання в процесі логування подій, мають зберігатися в гешованій формі. Також дана група вимог стандарту OWASP визначає те, що застосунок не має реєструвати конфіденційні дані користувачів відповідно до встановлених місцевою владою законів, при цьому, дана вимога вимагає дотримання правил реєстрації усіх подій, що пов’язані з безпекою: вдала та невдала автентифікація, логування, моніторинг, неправильно введений пароль тощо [34]. Дотримання даної вимоги при розробці веб-застосунку дозволить забезпечити якісний рівень безпеки застосунку.

Веб-застосунок використовує та передає велику кількість інформації, а це означає, що можлива бути втрата даних. Відповідно до цього у даному стандарті є вимога щодо загального захисту даних розробників та користувачів. Дана вимога підходить до усіх етапів розробки програмного забезпечення. Її дотримання повинно забезпечуватися регулярним копіювання важливих даних, відповідно до встановлених правил в політиці безпеки компанії, використання та реєстрація тестових даних, дотримання правил зберігання резервних копій, а також перевіркою того чи захищає програма конфіденційні дані від гешування сервером. Більше проаналізованих вимог наведено в ілюстративній частині 1 – Аналіз стандарту OWASP ASVS 4.0.3.

Усі відібрані вимоги дозволяють перевірити безпеку як стан веб-застосунку. Були відібрані також вимоги, які стосуються та можуть бути реалізовані саме для веб-застосунків. Кожна з цих вимог була відібрана відповідно прийнятих критеріїв відбору вимог, що наведені у пункті 2.3, які

відповідають певним етапам життєвого циклу розробки програмного забезпечення.

Таким чином усі відібрані вимоги можна сформувати у групи. Кожна з цих груп представляє собою обов'язкову перевірку, яка має бути здійснена, а відповідно до специфікацій та налаштувань тестованих веб-застосунків, дані вимоги можна буде декомпонизувати на декілька. Групування відібраних вимог наведено у таблиці 2.1.

Таблиця 2.1 – Груповані вимоги для методики перевірки безпеки як стану веб-застосунку

№	Вимога	Етап SDLC								Стандарт			
		1	2	3	4	5	6	7	8	ISO	NIST	COBIT	OWASP
1	Resource Planning	+	+							+		+	+
2	Requirements Analysis	+	+	+	+		+	+	+	+	+	+	+
3	SDLC/STLC Review		+	+		+	+	+	+	+		+	+
4	Design		+	+	+	+	+			+	+		+
5	Privacy Policy Analysis		+		+		+	+	+	+		+	+
6	Coding and Unit Testing					+	+				+		+
7	System Testing					+	+	+			+		+
8	Environment Review		+	+		+	+	+		+	+	+	+
9	Securing Files/Databases			+	+		+	+		+	+	+	+
10	WEB-application Testing					+	+	+	+	+			+

Проаналізувавши стандарти було виділено низку вимог, що стосуються безпеки застосунків, які можна перевірити в межах тестування безпеки як стану застосунку. Кожна з цих вимог відповідає підрозділу чи пункту свого стандарту. Використання стандартів допоможе розробити чек-лісти, тест-кейси та очікувані результати більш точними, а їх подальше застосування допоможе якісніше та швидше перевірити стан безпеки веб-застосунку.

2.5 Обґрунтування вибору інструментів для тестування

Для того, щоб тестування було якіснішим та швидшим, потрібно підібрати правильні інструменти для тестування. Після аналізу існуючих інструментів для тестування (підрозділ 1.3) було підібрано ряд застосунків, які будуть використані

під час тестування безпеки за розробленою методикою, а саме: Acunetix, Owasp Dependency-Check та Wireshark.

Acunetix допоможе просканувати веб-застосунок щодо вразливостей [23]. Серед його функціоналу буде звернуто увагу на сканування веб-ресурсів на предмет забутих веб-файлів та самих веб-застосунків. Розширений пошук вразливостей допоможе знайти ті, які навіть можуть бути відсутні при розробці, але при релізі самого застосунку можуть з’явитися. Також за допомогою нього буде просканово застосунок на предмет дії на нього різного роду атак, що надає Acunetix у своєму функціоналі.

Owasp Dependency-Check буде використаний для аналізу складу програмного забезпечення, або ж веб-застосунку, який може виявити публічні вразливості, що можуть знаходитися в інтеграціях з іншими різними застосунками або API [21]. За допомогою нього можна буде просканувати не тільки весь застосунок в цілому, але й його окремі компоненти. Даний інструмент був розроблений відносно стандарту OWASP, а тому результати його сканування будуть якіснішими.

Wireshark буде використаний для сканування трафіку пакетів, при роботі з веб-застосунком [24]. При виявлені пошкоджених чи підозрілих пакетів, буде проаналізовано їх вміст за допомогою функціоналу, що надає даний інструмент.

Також було підібрано ще один інструмент ImmuniWeb. Він виконує сканування обраного посилання на предмет вразливостей, API запитів, використання хмарного сховища, конфігурацію мережі, вразливостей OWASP та інше. Результат його роботи надасть змогу краще зрозуміти конфігурацію застосунку, навіть якщо немає доступу до коду.

Таким чином, було відібрано низку інструментів, що допоможуть здійснити тестування стану безпеки для веб-застосунків. Використання інструментів, які націлені на пошук вразливостей та пошкоджених даних, дозволять надати в майбутньому оцінку того, який стан безпеки є для сканованого веб-застосунку.

2.6 Висновки до розділу

Сформована математична модель опису демонструє процес розробки програмного забезпечення, як множину параметрів. Кожен з цих параметрів має вплив на загальний процес розробки застосунку. Створена модель допоможе якісніше здійснити оцінку процесу розробки під час застосування методики перевірки безпеки як стану веб-застосунку.

Проаналізувавши етапи життєвого циклу розробки веб-застосунків, було виділено критерії якості, яким повинні відповідати вимоги розроблюваної методики. Аналіз показав, що дотримання SDLC надасть змогу розробити застосунок за структурованими діями, але недотримання цієї структури може призвести до втрати безпеки як з функціонального боку так і з боку стану. При цьому кожна фаза SDLC є важливою, незалежно від розміру проекту та його цілі. Але дотримання критерій якості на усіх його етапах забезпечить якісний рівень безпеки. Висунутим критеріям повинні відповідати вимоги до методики.

Було проаналізовано низку стандартів щодо забезпечення безпеки програмного забезпечення та виділено вимоги для методики тестування безпеки як стану веб-застосунку, які базуються на застосуванні їх до перевірки веб-застосунку а також висунутих критеріїв якості. Сформовані вимоги були груповані, з метою їх подальшої декомпозиції відповідно до специфікацій веб-застосунку, що перевірятиметься. Після отримання вимог, яким повинна відповідати методика, можна переходити до розробки методики тестування безпеки веб-застосунків.

3 РОЗРОБКА МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

3.1 Розробка чек-ліста

Для ефективного застосування методики тестування безпеки для веб-застосунків, потрібно розробити чек-ліст. Чек-ліст – це перелік перевірок, які необхідно виконати [35]. На основі нього потім складаються тест-кейси. Для складання списку потрібно виділити необхідні для тестування компоненти.

Чек-ліст формується на основі висунутих вимог. По суті, кожен пункт чек-ліста це і є сама вимога, яку потрібно перевірити. Веб-застосунок може працювати на декількох браузерах, тому потрібно перевірити його стан безпеки на декількох браузерах. Список буде складений англійською мовою, осільки більша частина компонентів веб-застосунку має англійську назву. Основні перевірки наведені у таблиці 3.1.

Таблиця 3.1 – Чек-ліст основних перевірок

№	Перевірка	Environment		
		Google Chrome	Opera	Safari
S1	Resource Planning			
S2	Requirements Analysis			
S3	SDLC/STLC Review			
S4	Design			
S5	Privacy Policy Analysis			
S6	Coding and Unit Testing			
S7	System Testing			
S8	Environment Review			
S9	Securing Files/Databases			
S10	WEB-application Testing			

Розроблений чек-ліст містить не тільки вимоги, але й найпопулярніші браузери, які використовують користувачі у своїй роботі. При тестуванні веб-застосунку, неможна перевіряти лише на одному середовищі користування, оскільки помилки інтеграції одного браузера можуть не відображатися на іншому. Кожна з представлених вимог у таблиці 3.1 потребує декомпозиції перед початком тестування. Під час застосування методики, декомпозиція буде

відбуватися відносно тестованого веб-застосунку, адже наявність тих чи інших перевірок залежить від конфігурації застосунку.

Resource Planning (S1) допоможе правильно сформувати команду розробників та команду тестувальників, технічних спеціалістів та менеджерів. Без правильного використання ресурсів, при тестуванні чи розробці може виникнути така ситуація, коли якась функціональність була пропущена чи доступ був наданий не вірно. Саме тому грамотне планування ресурсів має бути здійснене відповідно до поставлених цілей розробки. Також на даному етапі визначаються ролі, які будуть використані під час тестування веб-застосунку, а саме: хакери (hackers) – роль зламу доступу, зломщики (crackers) – роль зламу застосунку, етичні хакери (ethical hacker) – роль зламу доступу із потрібним дозволом, хаотичний злам (packet monkeys) – роль зламу доступу зі знанням коду. Кожна компанія обирає відповідну людину на кожну роль.

Requirements Analysis (S2) надає можливість проаналізувати усі вимоги, які існують для розроблюваного веб-застосунку. До них відносяться: business, stakeholder, solution, functional, non-functional, transition, design, system, user, API, definition of done criteria, та інші. Також, потрібно проаналізувати так звані -bility requirements: usability, security, reliability, performance, availability, scalability, accessibility. Тестування вимог дуже важливий етап в розроблюваній методиці, адже вимог є досить велика кількість, буває таке що одна вимога суперечить іншій, а тому при розробці веб-застосунку можуть бути пропущені важливі пункти, які стосуються безпеки.

SDLC/STLC (S3) – описують цикл розробки та тестування застосунків відповідно. Грамотне планування циклу розробки – забезпечить грамотне планування циклу тестування. Якщо буде пропущено, наприклад, фаза збору вимог, то в майбутньому веб-застосунок може бути розроблений без важливих функціональних та нефункціональних вимог щодо безпеки.

Design (S4) включає в себе аналіз ризиків та загроз безпеці розроблюваного веб-застосунку. Даний пункт загальних перевірок також

включає в себе розробку плану тестування ризиків, який повинен включати в себе фазу тестування безпеки.

Privacy Policy Analysis (S5) є також однією із найважливіших перевірок. Якщо компанія - розробник має політику безпеки або ж інший документ, де містяться правила користування апаратним та програмним забезпеченням, правила налаштування та використання, правила доступу та зміни інформації, яка циркулює всередині компанії, то такий документ має обов'язково бути протестованим. Для подальшого тестування потрібно знати не тільки те, яка інформаціє є конфіденційною, але й хто та який має до неї доступ, чи є відповідні системи логування подій та чи взагалі ведеться моніторинг трафіку інформації всередині компанії.

Coding and Unit Testing (S6) включає в себе різного роду перевірки. Зокрема обов'язково має бути зроблено client-server architecture analysis та code review. Аналіз архітектури застосунку дозволить побачити, які компоненти при його розробці були пропущені, які є надлишковими, які взагалі не відповідають вимогам, тобто усі ті компоненти які окремо або в цілому мають вплив на безпеку застосунку. Code review має здійснюватися на усіх етапах розробки застосунку від початку написання першої стрічки коду. При неграмотному побудові коду чи описі функцій, зловмисник матиме змогу дістатися даних, які обробляються застосунком.

System Testing (S7) включає в себе ряд перевірок, які відносяться до конфігурації операційної системи де розробляється застосунок. Також, має бути здійснена перевірка усіх привілеїв та доступів системи.

Environment Review (S8) та перевірка, яка включає в себе перевірку усього програмного забезпечення, що використовується при розробці та тестуванні. Тобто має бути перевірено конфігурацію, налаштування, доступ до даних системи, доступ до даних інших застосунків, ліцензійність, наявність шкідливого програмного забезпечення та інше. Також особливу увагу варто приділяти програмному забезпеченню яке стосується тестування, адже при тестуванні можуть використовуватися різні дані, і при неправильному використанні

відповідних інструментів, дані веб-застосунку або ж його функціональність може бути пошкоджена, що досить сильно впливає на безпеку застосунку в цілому.

Дана перевірка також включає в себе перевірку HTTPS веб-застосунку. Тобто має бути перевірено склад посилання, параметри (якщо такі є) що передаються в посиланні, запити, які проходять через застосунок, сервер, заголовки, response, контент-тип та інше.

Securing Files/Databases (S9) перевірки, які включають в себе перевірки інформації: тип, доступ застосунків розробника та тестувальника, доступ користувачів, використання, зміну, знищення беззворотнє, видалення та інше.

WEB-application Testing (S10) перевірки, які стосуються безпосередньо тестування останньої версії веб-застосунку. Має бути перевірено весь функціонал застосунку, дизайн, нефункціональні вимоги, а особливу увагу варто приділяти тим функціям, які використовують дані користувача або ж передають їх.

Таким чином було сформовано перелік основних перевірок. Даний чек-ліст був сформований відповідно до відібраних вимог проаналізованих стандартів, ризиків та загроз. Пункти даного чек-ліста відповідають сформованим вимогам, що наведені у розділі 2.3. Кожна з цих перевірок вимагає декомпозиції, але це вже має відбуватися відповідно до веб-застосунку, для якого цей чек-ліст буде застосовано. Звичайно, дані перевірки можна змінювати, додавати або ж деякі викреслювати, це все залежить від специфікації тестованого застосунку, але чек-ліст було складено таким чином, щоб усі найголовніші перевірки були покриті.

3.2 Розробка тест-кейсів

Основою для розробки тест-кейсів буде розроблений чек-ліст (табл. 3.1). Кожен пункт чек-ліста може мати декілька перевірок. Дані перевірки оформлюються у тест-кейси [36]. Для їх оформлення будуть використані тест-кейси high-level типу. Даний тип дозволяє сформувати кейси таким чином, щоб

суть перевірки була сформована в одне речення достатньо коротко, наскільки це можливо, але в той час достатньо зрозуміло.

Оскільки розробка методики передбачає її використання під час розробки веб-застосунку та після, буде написано ряд тест-кейсів для останнього пункту чек-ліста, а саме WEB-application Testing.

Розроблені текст-кейси, що покривають основні перевірки наведено у таблиці 3.2.

Таблиця 3.2 – Тест-кейси основних перевірок для пункту WEB-application Testing

№	Пункт чек-ліста	Тест-кейс	Статус
1	S10	Перевірити відображення усієї інформації веб-застосунку	
2	S10	Перевірити конфігурацію доступів веб-застосунку	
3	S10	Перевірити чи мережа та комп’ютер використовують брандмауер	
4	S10	Перевірити, що конфіденційні дані не передаються через URL	
5	S10	Перевірити через маніпулювання URL адресою, що застосунок не відображає небажану інформацію	
6	S10	Перевірити, що паролі зашифровані та передаються безпечно, наприклад, через HTTPS	
7	S10	Перевірити інформацію, яка зберігається в куках та кеші (повинна мати читабельний вигляд)	
8	S10	Перевірити, що вся конфіденційна інформація користувача та застосунку замаскована	
9	S10	Перевірити, що веб-застосунок жорстко не запрограмований на ім’я користувача чи його пароль	
10	S10	Перевірити, що введений пароль відповідає стандартним вимогам (8 length, великі/малі літери, цифри, символи)	
11	S10	Перевірити, що функція скидання паролю є безпечною	
12	S10	Перевірити, що персональні дані користувача не використовуються в нормальному вигляді	
13	S10	Перевірити, що функція запам’ятовування даних не використовує авто-заповнення доля паролів та фінансових даних	
14	S10	Перевірити, що веб-застосунок захищений щодо несанкціонованого доступу	
15	S10	Перевірити, що застосунок дозволяє завантажувати файли, перевіряючи їх при цьому через антивірус	
16	S10	Перевірити, що застосунок не використовує відкритий порт	
17	S10	Перевірити конфігурацію мережі (Wi-Fi)	

Продовження таблиці 3.2

№	Пункт чек-ліста	Тест-кейс	Статус
18	S10	Перевірити запити HTTPS, які використовує застосунок (PUT/DELETE запити заборонені до використання)	
19	S10	Перевірити, що сторінка логіну та паролю має містити інформативні повідомлення про помилки	
20	S10	Перевірити, що збій в роботі веб-застосунку не повинен викликати відображення конфіденційної інформації	
21	S10	Перевірити, що деталі внутрішньої системи не мають бути ні в жодному з повідомлень про помилки	
22	S10	Перевірити записи логування (конфіденційна інформація не повинна бути там відображена)	
23	S10	Перевірити, що журнали логування подій ведуться з відповідними доступами	
24	S10	Перевірити, що критичні ресурси веб-застосунку мають бути доступні тільки авторизованим користувачам та службам	
25	S10	Перевірити, що сеанс користувача закінчується після виходу чи закриття веб-застосунку	
26	S10	Перевірити, що використовується актуальна версія веб-застосунку	
27	S10	Перевірити, що веб-застосунок захищений до атак формату введення даних	
28	S10	Перевірити, що веб-застосунок захищений до атак типу грубої сили	

Розроблені тест-кейси покривають не всі можливі перевірки, а лише найголовніші. Кожен тест-кейс можна розділити на декілька, це залежить від конфігурації веб-застосунку. Тест-кейси були розроблені на основі відібраних вимог проаналізованих стандартів. Кожен з цих тест-кейсів можна буде перетворити у low-level вигляд, тобто такий, де чітко буде прописаний кожен крок виконання тест-кейсу, але без останньої актуальної робочої версії веб-застосунку, таке перетворення здійснити неможливо. Після виконання перевірок має бути поставлений відповідних статус: Passed або Failed. Цей статус надає можливість зрозуміти, який з кейсів був успішно пройдений, а який ні. Після формування основного чек-ліста та деяких тест-кейсів, можна переходити до розробки очікуваних результатів.

3.3 Розробка очікуваних результатів

Очікувані результати – це ті результати, які мають бути досягненні при виконанні відповідних тест-кейсів. Розробляти точні очікувані результати не доцільно, а тому буде здійснена оцінка того, наскільки кожна вимога має бути перевірена тест-кейсами. Адже кожен веб-застосунок є унікальним, а їх кількість перевищує тисячі, тому важко передбачити в якому є та чи інша функціональність, а в якому - ні.

Для розробки очікуваних результатів було здійснено експертне оцінювання за методом експертної оцінки. Для цього було обрано групу експертів різного рівня, які відносяться до сфери тестування та тестування безпеки. Їм було надано перелік вимог розроблюваної методики та запропоновано здійснити оцінку необхідності кожної з вимог.

Експерти:

E1 – Експерт 1, рівень Junior;

E2 – Експерт 2, рівень Middle;

E3 – Експерт 3, рівень Senior.

Так як вимоги були згруповані у 10 пунктів, а експертів є лише 3, складемо таблицю в яку буде занесено оцінки експертів відповідно до кожної вимоги. Таким чином, агрегація отриманих оцінок відбуватиметься відповідно формули:

$$Q_p = \frac{\sum B_i}{10}, \text{ де} \tag{3.1}$$

Q_p – середня оцінка кожного експерта щодо всіх вимог;

B_i – оцінка кожної вимоги (B_i).

$$S_i = \frac{\sum B_{ij}}{3}, \text{ де} \tag{3.2}$$

S_j – середня оцінка кожної вимоги;

B_{ij} – оцінка кожної вимоги (i) відповідного кожного експерта (j).

Середня оцінка кожного експерта щодо всіх вимог означає середню оцінку всім вимогам одразу від одного з експертів. Для подальшого аналізу результатів буде використана саме середня оцінка кожної вимоги, так як експерти різного

рівня зможуть оцінити її кожен по-своєму, будуть отримані результати необхідності кожної з вимог для розроблюваної методики.

Розрахунок оцінок наведено у таблиці 3.3.

Таблиця 3.3 – Розрахунок оцінки необхідності кожної вимоги

	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	Q_p
E1	0.7	0.85	0.7	0.7	0.8	0.85	0.8	0.7	0.8	0.85	**0.775**
E2	0.85	0.9	0.85	0.9	0.85	0.9	0.8	0.85	0.8	0.85	**0.855**
E3	0.9	1	0.9	1	0.9	1	0.95	0.9	0.9	1	**0.945**
S_j	**0.82**	**0.92**	**0.82**	**0.87**	**0.85**	**0.92**	**0.85**	**0.82**	**0.83**	**0.9**	

Розрахунок необхідності кожної вимоги показав, що найкращу оцінку надав Експерт 3. Найбільшу оцінку серед вимог отримали вимоги B2, B6, B10.

Вимога B2 отримала оцінку – 0.92, що означає – здійснення перевірки, тобто тестування даної вимоги, при тестуванні безпеки веб-застосунку має бути здійснена на 92%. Вимоги є важливим аспектом при розробці будь-якого програмного забезпечення, а тому вони мають бути ретельно проаналізовані та протестовані на відповідність до встановлених критеріїв. Дана вимога має відповідати критеріям майже усіх етапів SDLC, також перевірка її доцільності має здійснюватися відповідно до відібраних раніше вимог кожного зі стандартів.

Вимога B6 отримала оцінку – 0.92, що означає те, що вона має бути протестована не менше ніж на 92%. Дана вимога відноситься до критеріїв якості відібраних для п’ятого та шостого етапів SDLC. Забезпечення якості реалізації цієї вимоги може усунути безліч недоліків при написанні коду та проведенні його unit тестуванні. Адже гарно написаний код, який має свою зрозумілу структуру, без зайвих чи неприйнятих вимог щодо його написання, забезпечить якісніший аналіз самого коду та прозорість його написання для розробників, що впливає на реалізацію методів захисту.

Вимога B10 отримала оцінку – 0.9. Дана вимога описує низку критеріїв, які відносяться до останніх чотирьох етапів SDLC. Забезпечення правильності реалізації цієї вимоги, тобто правильного плану тестування, правильної та коректної перевірки функціоналу та наповнення веб-застосунку, дозволить отримати якісні дані про стан застосунку в цілому, а вимоги, які забезпечують

безпеку застосунку будуть перевірятися в сукупності з іншими не функціональними вимогами. Результатом такого тестування стане звіт про якість веб-застосунку, а також звіти дефектів, якщо такі будуть знайдені.

Найнижчу оцінку серед усіх вимог отримали вимоги В1, В3, В8. Вимога В1 повинна бути реалізована відповідно до встановлених критеріїв перших двох етапів SDLC. Забезпечення команди розробників, де кожен матиме свою відповідну роль, надає можливість структуровано розробляти веб-застосунок. Дана вимога отримала одну із найнижчих оцінок, це можна пояснити тим, що склад команди дуже часто змінюється, а тому забезпечення цієї вимоги більш ніж на 90% іноді буває навіть неможливим.

Вимога В3 повинна бути реалізована відповідно до встановлених критеріїв майже усіх етапів SDLC. Наразі багато хто лише говорить про життєвий цикл розробки програмного забезпечення, але дотримання усіх його етапів та встановлений критеріїв якості відбувається далеко не завжди. Стосовно веб-застосунків, то іноді взагалі забувають про останній етап, хтось об’єднує декілька етапів в один, а тому забезпечення повноти цієї вимоги важко відслідкувати.

Вимога В8 повинна бути реалізована відповідно до встановлених критеріїв якості етапів розробки та реалізації SDLC. Коли необхідно терміново протестувати якийсь функціонал чи щось інше, часто забувається про те, яке середовище чи інструменти будуть для цього використані. Постійна підтримка та розвиток веб-застосунків несе за собою постійне оновлення технологій, середовищ розробки та тестування. А тому, при появі якогось оновлення середовища програмування, чи тестування, чи API не завжди звертають увагу що саме було змінено, та як це вплине на безпеку розроблюваного веб-застосунку. Тому забезпечити виконання даної вимоги більше ніж на 90% буде дуже складно у зв’язку з постійним оновленням стеку технологій.

Таким чином, було здійснено оцінку очікуваних результатів методом експертної оцінки важливості кожної вимоги, тобто наскільки кожна вимога може бути покрита чек-лістом та тест-кейсами.

3.4 Висновки до розділу

Розробка методики тестування передбачає розробку чек-лістів та тест-кейсів. Відібрані вимоги були проаналізовані згруповані у 10 окремих вимог. Кожна з них відноситься до різного роду веб-застосунків та зможе бути перевірена відповідними текст-кейсами. При створенні чек-ліста було обрано список найпопулярніших браузерів, які використовуються в роботі для веб-застосунків.

Відповідно до розробленого чек-ліста було сформовано низку тест-кейсів, які перевіряють повноту реалізації вимоги WEB-application testing. Було обрано саме цю вимогу для створення тест-кейсів, оскільки розробка самого застосунку не ведеться, а тому перевірити попередні вимоги є неможливим. Сформований перелік тест-кейсів покриває основні перевірки, які мають бути здійснені відповідно до реалізації даної вимоги. Звісно, створення тест-кейсів відбувається окремо під кожен веб-застосунок, так як розміщення тих самих кнопок є різним, а тому кроки до виконання перевірки будуть різними.

Було сформовано оцінювання необхідності реалізації кожної з вимог. Оцінка показала, що найбільше потрібно реалізувати вимоги В2, В6 та В10, що відносяться до відповідних пунктів складеного чек-ліста. Найменшу оцінку отримали вимоги В1, В3, В8. Було обґрунтовано величину кожної оцінки.

Після формування чек-ліста, тест-кейсів та реалізації очікуваних результатів можна переходити до безпосереднього тестування стану безпеки веб-застосунків.

4 ЗАСТОСУВАННЯ МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ ДЛЯ ВЕБ-ЗАСТОСУНКІВ

4.1 Застосування на прикладі веб-застосунків з постачання послуг

Для перевірки коректності методики тестування стану безпеки веб-застосунка, було обрано веб-застосунок з надання державних послуг онлайн «Дія» [37]. Осільки даний застосунок використовує багато користувачів, а кількість конфіденційної та важливої інформації, що передається застосунком досягає великих масштабів, то перевірити стан безпеки даного веб-застосунку буде гарним прикладом для перевірки коректності розроблюваної методики.

Під час розробки методики тестування безпеки веб-застосунків було реалізовано низку тест кейсів. Кожен тест кейс відповідає вимозі WEB-application testing, яка включає в себе перевірку вимог, що стосуються вже розроблюваного застосунку. Відповідно до цього, кожен тест-кейс має свій статус. Статус Pass означає що, тест-кейс пройдено успішно та не було ніяких виявлених дефектів. Статус Fail означає що тест-кейс не було пройдено до кінця або ж було заблоковано по причині відсутності потрібних даних.

Таблиця 4.1 – Результат проходження тест-кейсів

№	Пункт чек-ліста	Тест-кейс	Статус
1	S10	Перевірити відображення усієї інформації веб-застосунку	Pass
2	S10	Перевірити конфігурацію доступів веб-застосунку	Fail
3	S10	Перевірити чи мережа та комп’ютер використовують брандмауер	Pass
4	S10	Перевірити, що конфіденційні дані не передаються через URL	Pass
5	S10	Перевірити через маніпулювання URL адресою, що застосунок не відображає небажану інформацію	Pass
6	S10	Перевірити, що паролі зашифровані та передаються безпечно, наприклад, через HTTPS	Pass
7	S10	Перевірити інформацію, яка зберігається в куках та кеші (повинна мати читабельний вигляд)	Pass
8	S10	Перевірити, що вся конфіденційна інформація користувача та застосунку замаскована	Pass
9	S10	Перевірити, що веб-застосунок жорстко не запрограмований на ім’я користувача чи його пароль	Fail

Продовження таблиці 4.1

№	Пункт чек-ліста	Тест-кейс	Статус
10	S10	Перевірити, що введений пароль відповідає стандартним вимогам (8 length, великі/малі літери, цифри, символи)	Fail
11	S10	Перевірити, що функція скидання паролю є безпечною	Fail
12	S10	Перевірити, що персональні дані користувача не використовуються в нормальному вигляді	Pass
13	S10	Перевірити, що функція запам’ятовування даних не використовує авто-заповнення доля паролів та фінансових даних	Pass
14	S10	Перевірити, що веб-застосунок захищений щодо несанкціонованого доступу	Pass
15	S10	Перевірити, що застосунок дозволяє завантажувати файли, перевіряючи їх при цьому через антивірус	Fail
16	S10	Перевірити, що застосунок не використовує відкритий порт	Pass
17	S10	Перевірити конфігурацію мережі (Wi-Fi)	Pass
18	S10	Перевірити запити HTTPS, які використовує застосунок (PUT/DELETE запити заборонені до використання)	Pass
19	S10	Перевірити, що сторінка логіну та паролю має містити інформативні повідомлення про помилки	Pass
20	S10	Перевірити, що збій в роботі веб-застосунку не повинен викликати відображення конфіденційної інформації	Pass
21	S10	Перевірити, що деталі внутрішньої системи не мають бути ні в жодному з повідомлень про помилки	Pass
22	S10	Перевірити записи логування (конфіденційна інформація не повинна бути там відображена)	Pass
23	S10	Перевірити, що журнали логування подій ведуться з відповідними доступами	Fail
24	S10	Перевірити, що критичні ресурси веб-застосунку мають бути доступні тільки авторизованим користувачам та службам	Pass
25	S10	Перевірити, що сеанс користувача закінчується після виходу чи закриття веб-застосунку	Pass
26	S10	Перевірити, що використовується актуальна версія веб-застосунку	Pass
27	S10	Перевірити, що веб-застосунок захищений до атак формату введення даних	Pass
28	S10	Перевірити, що веб-застосунок захищений до атак типу грубої сили	Pass

При перевірці безпеки веб-застосунка шляхом проходження тест-кейсів було пройдено не всі тест-кейси, деякі з них мають статус Fail. Для отримання якіснішого результату тестування, обраний веб-застосунок буде протестовано через низку допоміжних інструментів.

Тестування через інструмент ImmuniWeb [38] надає можливість перевірити веб-застосунок на предмет наявних вразливостей, а звіт, який буде отримано після закінчення роботи даного інструменту можна буде проаналізувати відповідно до того, які вимоги та критерії були перевірені, та здійснити кращу оцінку стану безпеки веб-застосунку (рис. 4.1).

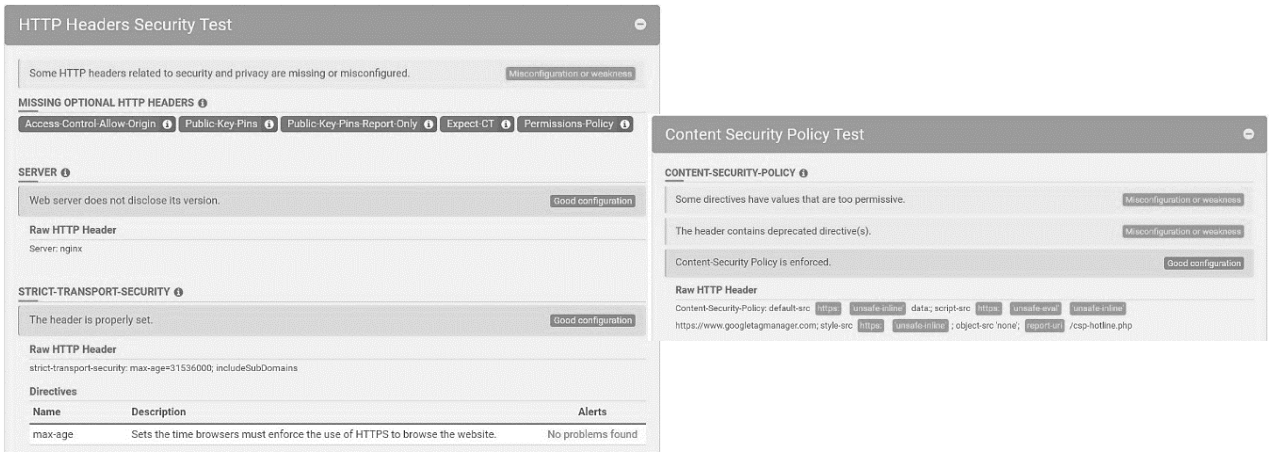


Рисунок 4.1 – Результати тестування веб-застосунку за допомогою ImmuniWeb

Для проведення тестування на предмет вразливостей та дотримання вимог діючих та раніше описаних стандартів безпеки, було обрано інструмент Acunetix (рис.4.2).

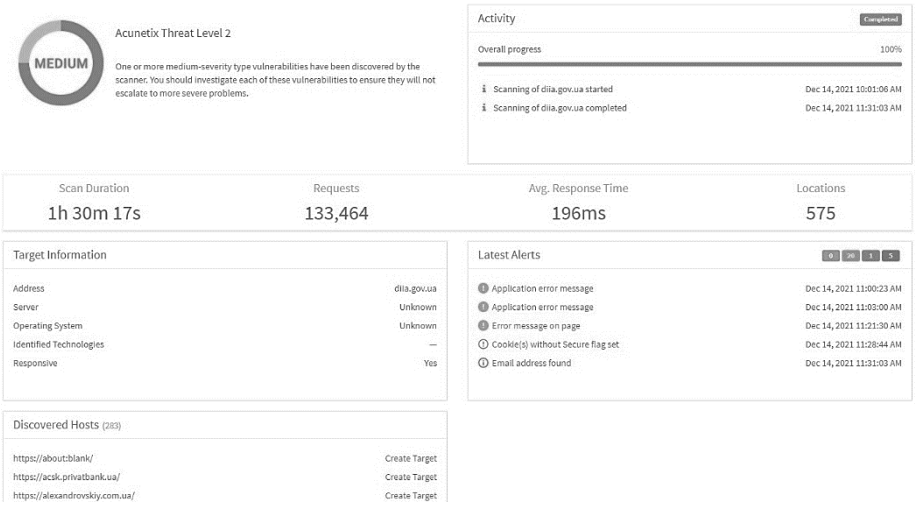


Рисунок 4.2 – Результати тестування застосунком Acunetix

Для отримання якіснішого результату було обрано вимоги щодо отриманих результатів: має бути перевірено увесь веб-застосунок, звіт про

сканування безпеки веб-застосунку має бути складений відповідно до встановлених вимог діючих стандартів з безпеки. Таке налаштування сканування дозволить провести повне тестування веб-застосунку із використанням великої кількості запитів різного типу, що надасть детальний звіт про стан безпеки веб-застосунку.

При проведенні тестування даним застосунком було виділено також низку вразливостей, які потрібно просканувати. Результат такого сканування наведено на рисунку 4.3.

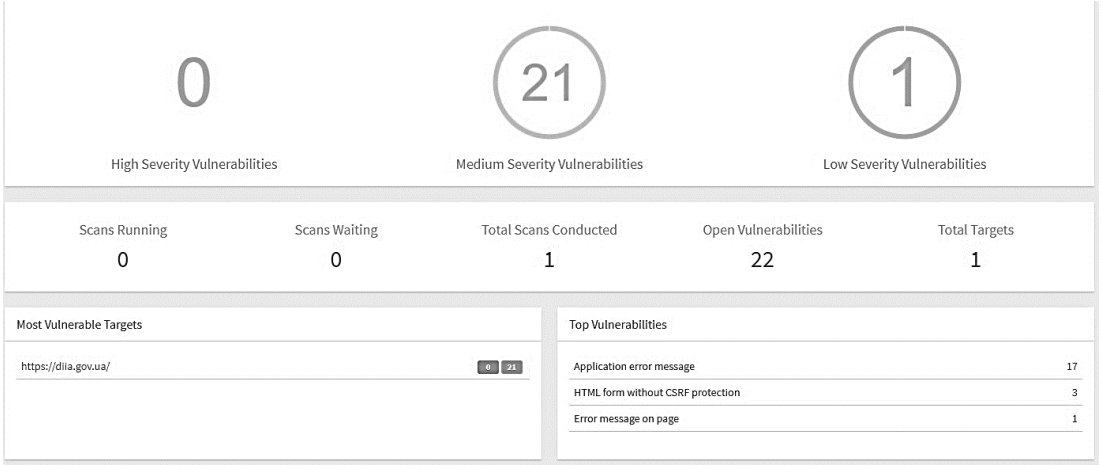


Рисунок 4.3 – Результат сканування вразливостей веб-застосунку «Дія»

Більше результатів тестування наведено в ілюстративній частині 7. Після отримання результатів тестування можна переходити до оцінки отриманих результатів, порівнюючи з тим, які були розроблені для даної методики тестування.

4.2 Аналіз отриманих результатів

З результатів проведення перевірки коректності застосування методики тестування безпеки веб-застосунків, було виявлено, що не всі розроблені тест-кейси були пройдені успішно.

Тест-кейс під номером 2 має статус Fail. Даний тест-кейс є заблокованим, оскільки перевірити конфігурацію доступів до застосунку з боку звичайного користувача немає можливості.

Тест-кейси під номером 9, 10, 11 мають статус Fail. Дані тест-кейси є також заблокованими, оскільки реєстрація та здійснення входу до особистого кабінету користувача, де він може побачити свої документи відбувається через сторонні застосунки входу, зокрема через підтвердження особистості через обраний користувачем банк. Тому здійснити перевірку чи є веб-застосунок жорстко запрограмований на ім’я користувача та пароль, чи перевірку валідації паролю, чи перевірку функції скидання паролю неможливо.

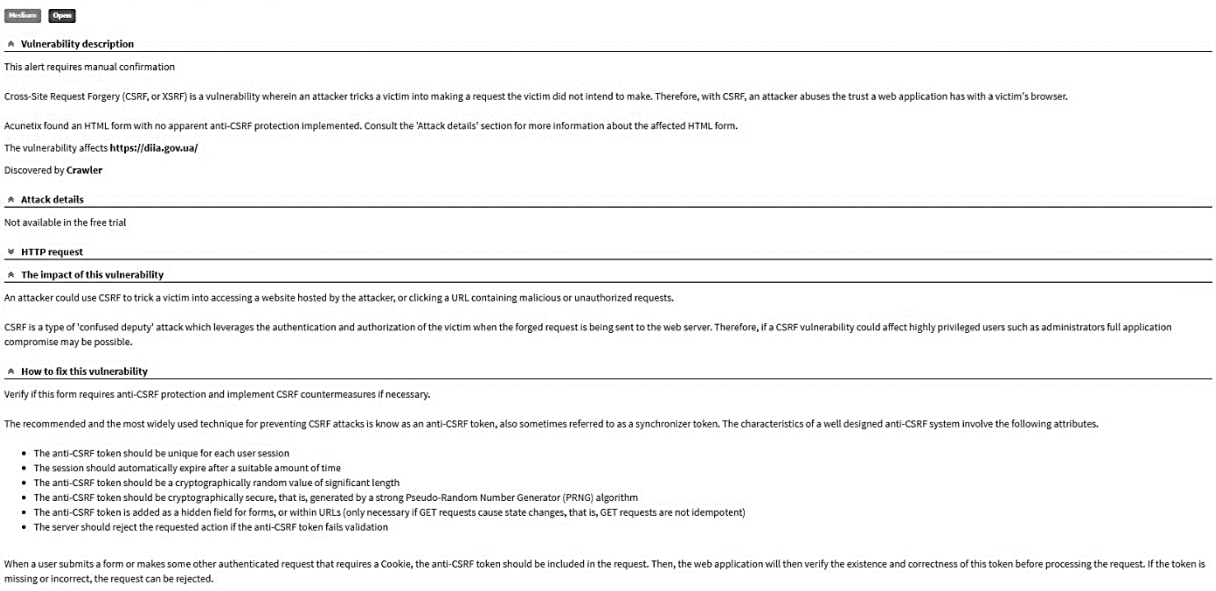
Також тест-кейс під номером 15 був заблокований, осільки веб-застосунок не дозволяє завантажувати особисті дані користувача напряму із девайсів користувача. Усі документи чи особиста інформація користувача береться із бази даних, та при підтвердженні його особистості, синхронізуються з особистим кабінетом.

Тест-кейс під номером 23 був заблокований по причині неможливості виконання. Звісно при тестуванні було виконано логування подій, але лише як зі сторони користувача, що можуть зробити усі ті, хто має відповідні інструменти. Але перевірка саме того, що журнали логування подій ведуться із відповідними доступами перевірити неможливо, так як до конфігурації доступів даного застосунку немає прямого доступу.

Таким чином тест-кейсами було перевірено більшу частину стану безпеки веб-застосунку. Для детальнішого аналізу було проведено тестування допоміжними раніше відібраними інструментами перевірки стану безпеки веб-застосунку.

Результат сканування за допомогою інструменту ImmuniWeb показав, що застосунок має вразливі місця в наявному HTTP header. Звісно ці вразливості є не критичними з одного боку, так як їх суть у тому, що використовуються дещо застарілі конфігурації та налаштування заголовків. Але з іншого боку, це може стати гарним інструментом для зловмисника отримати чи підмінити версію сайту. Також було виявлено, що дана вразливість може впливати на XXS атаки з боку зловмисника. Для більш детальної інформації було проведено тестування за допомогою інструменту Acunetix.

Даний інструмент також виявив вразливість у напрямку HTTP. Але якщо перший застосунок показав лише наявні та проблемні місця, то другий було налаштовано таким чином, щоб звіт формувався за на основі аналізі цих вразливостей відповідно до стандартів ISO, NIST, OWASP. Також при аналізі вразливостей даний інструмент надав не тільки детальну інформації щодо того як і де знаходиться вразливість, але й рекомендації щодо її усунення (рис. 4.4).


Medium Open

Vulnerability description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

The vulnerability affects **https://diia.gov.ua/**

Discovered by **Crawler**

Attack details

Not available in the free trial

HTTP request

The impact of this vulnerability

An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

CSRF is a type of 'confused deputy' attack which leverages the authentication and authorization of the victim when the forged request is being sent to the web server. Therefore, if a CSRF vulnerability could affect highly privileged users such as administrators full application compromise may be possible.

How to fix this vulnerability

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Рисунок 4.4 – Рекомендації щодо усунення вразливості HTTP/HTML

Таким чином можна зробити оцінку коректності застосування методики тестування безпеки веб-застосунків, а саме – методика є ефективним способом перевірки стану безпеки веб-застосунку. Із розробки очікуваних результатів вимога, яка перевірялася, мала бути перевірена не менше ніж на 90%. З урахуванням того, що деякі тест-кейси були заблоковані з причини неможливості їх виконання, так як та чи інша функціональність є відсутньою, а інші вимоги були перевірені за допомогою відповідних застосунків, можна зробити висновок, що дана вимога була перевірена більше ніж на 90%. За отриманими результатами тестування можна розробити план щодо мінімізації та усунення виявлених вразливостей, так як було отримано не тільки їх аналіз, але й рекомендації щодо їх усунення.

4.3 Висновки до розділу

Для перевірки коректності застосування методики тестування безпеки для веб-застосунків було обрано застосунок з надання державних послуг онлайн «Дія». Даний застосунок перевірявся за допомогою проходження розроблених тест-кейсів та додатковим застосуванням інструментів сканування стану безпеки.

На основі отриманих результатів було зроблено висновок, що застосування розробленої методики для перевірки стану безпеки є ефективним способом здійснення цієї перевірки. Адже розроблені тест-кейси покривають основні перевірки та можуть виявити низку вразливостей. До того ж, при перевірці стану безпеки дійсно, багато що можна перевірити за допомогою відповідних застосунків, але для перевірки вимог щодо менеджменту забезпечення безпеки, потрібно використовувати наведені вище тест-кейси.

Таким чином на основі отриманих результатів тестування перевірка стану безпеки веб-застосунку була здійснена більше ніж на 90%, що відповідає розробленим очікуваним результатам.

5 ЕКОНОМІЧНА ЧАСТИНА

Метою економічної частини магістерської кваліфікаційної роботи є обґрунтування економічної доцільності розробленої методики тестування безпеки веб-застосунків. Для виконання поставленої мети необхідно:

- оцінити комерційний потенціал розробки;
- оцінити витрати на виконання та впровадження результатів наукової роботи;
- розрахувати ціну та чистий прибуток реалізації результатів розробки;
- розрахувати період окупності наукової роботи та результатів розробки.

5.1 Оцінювання комерційного потенціалу розробки (технологічний аудит розробки)

Об’єктом дослідження магістерської кваліфікаційної роботи є процес тестування безпеки веб-застосунків.

Було залучено трьох незалежних експертів для проведення технологічного аудиту, а саме: Баришев Ю. В., Куперштейн Л. М., Войтович О. П. Кожному експерту була надана таблиця, яку він повинен заповнити своїми оцінками після ознайомлення із запропонованою розробкою. Дана таблиця визначатиме критерії оцінювання потенціалу розробки та їх можливу оцінку в балах. Після виконання цього, буде обрахована середньоарифметична сума балів та буде визначено рівень економічного потенціалу розробки. Критерії оцінювання потенціалу розробки наведено у додатку Б.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В.	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
1	2	2	2
2	3	3	3
3	4	4	4
4	4	3	3
5	4	3	3
6	3	3	3

Продовження таблиці 5.1

Критерії	Прізвище, ініціали експерта		
	Баришев Ю. В.	Куперштейн Л. М.	Войтович О. П.
	Бали, виставлені експертами:		
7	2	3	3
8	4	3	3
9	4	3	4
10	4	3	4
11	3	4	4
12	3	3	3
Сума балів	$СБ_1$=40	$СБ_2$=37	$СБ_3$=39
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^i СБ_i}{i} = \frac{40+37+39}{3} = 38{,}6$		

За отриманими результатами оцінки із таблиці 5.1, можна зробити висновок базуючись на таблиці 5.2 рівнів комерційного потенціалу розробки.

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів	Рівень комерційного потенціалу
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-50	Високий

Відповідно до таблиці 5.2, рівень комерційного потенціалу розробки методики тестування безпеки для веб-застосунків вище середнього.

Оцінювання рівня якості розробки методики тестування безпеки для веб-застосунків здійснюється з метою порівняльного аналізу та визначення найбільш ефективного, з технічної точки зору, варіанта інженерного рішення.

Рівень якості – кількісна характеристика міри придатності певного виду продукції для задоволення конкретного попиту на неї при порівнянні з відповідними базовими показниками за фіксованих умов споживання.

Абсолютний рівень якості розробки методики тестування безпеки веб-застосунків потрібно знайти за обчисленням обраних для вимірювання показників, при цьому не порівнюючи їх із відповідними показниками

аналогічних засобів. Для цього було обрано низку параметрів: ймовірність визначення вразливості, обсяг даних для перевірки, кількість взаємодій між компонентами одного застосунку. Визначаємо величину параметрів якості в балах та встановлюємо граничні значення (гірші, кращі, середні). Дані для кожного параметра представлено у таблиці 5.3.

Таблиця 5.3 – Основні параметри методики тестування безпеки для веб-застосунків

Параметри	Абсолютне значення параметру			Коефіцієнт вагомості параметру
	Краще +5…+4	Середнє +3	Гірше +1…+2	
Ймовірність визначення вразливостей	4			0,6
Обсяг даних при перевірці		3		0,3
Кількість взаємодій між компонентами одного застосунку			2	0,2

Із врахування коефіцієнтів вагомості відповідних параметрів можна визначити абсолютний рівень якості інноваційного рішення за формулою:

$$K_{я.а.} = \sum_{i=1}^{n} Рн_i * a_i, \text{ де} \quad (5.1)$$

$Рн_i$ – числове значення і-го параметра інноваційного рішення, n – кількість параметрів інноваційного рішення, що прийняті для оцінювання, аі – коефіцієнт вагомості відповідного параметра (сума коефіцієнтів вагомості всіх параметрів повинна дорівнювати 1). Таким чином, абсолютний рівень якості методики тестування безпеки веб-застосунків складає 3.7 бали. Тепер потрібно визначити відносний рівень якості та порівняти показники з абсолютними показниками якості найліпших аналогів (табл. 5.4).

Таблиця 5.4 – Порівняння основних параметрів методики тестування безпеки веб-застосунків

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметру
	Базовий	Новий		
Ймовірність визначення вразливостей	0.6	0.7	3	0.6

Продовження таблиці 5.4

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметру
	Базовий	Новий		
Обсяг даних при перевірці	80мв	95мв	2.5	0.3
Кількість взаємодій між компонентами одного застосунку	12	25	1	0.2

Відносний рівень якості методики тестування безпеки веб-застосунків визначаємо за формулою 5.2.

$$K_{я.в.} = \sum_{i=1}^{n} q_i * a_i, \text{ де} \quad (5.2)$$

Відносний рівень для методики тестування безпеки веб-застосунків становить 2.75 – тобто розроблена методика якісніша на 74%.

Конкурентоспроможність визначається критеріям оцінювання споживачів конкретного ринку. Для розрахунку конкурентоспроможності необхідно сформувати таблицю 5.5, із відповідними показниками.

Таблиця 5.5 – Нормативні та економічні параметри методики тестування безпеки веб-застосунків.

Параметри	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметру
	Базовий	Новий		
Ймовірність визначення вразливостей	0.6	0.7	3	0.6
Обсяг даних при перевірці	80мв	95мв	2.5	0.3
Кількість взаємодій між компонентами одного застосунку	12	25	1	0.2
Ціна за продукт	19000	15000	0.75	-

Загальний показник конкурентоспроможності (К) визначається з урахуванням вищевказаних параметрів за формулою 5.3.

$$K = \frac{I_{Т.П.}}{I_{Е.П.}} = \frac{2,75}{0,79} = 3,5, \quad (5.3)$$

де $I_{Т.П.}$ – індекс технічних параметрів (відносний рівень якості інноваційного рішення); $I_{Е.П.}$ – індекс економічних параметрів розрахований нижче за формулою 5.4:

$$I_{Е.П.} = \frac{РН_{ЕІ}}{РБ_{ЕІ}} = \frac{15000}{19000} = 0,79 \quad (5.4)$$

де $РН_{ЕІ}$, $РБ_{ЕІ}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

Відповідно до розрахунків загальний показник конкурентоспроможності становить 3.5, що свідчить про більшу конкурентоспроможність аніж в порівнянні з аналогом.

5.2 Прогнозування витрат на виконання науково-дослідної та конструкторсько-технологічної роботи

5.2.1 Розрахунок витрат, що стосуються виконавців розробки методики тестування безпеки веб-застосунків

Команда розробки методики тестування безпеки веб-застосунків складається із керівника, розробника та тестувальника.

Основна заробітна плата для розробників (дослідників) *Зо*, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{M}{T_p} t \qquad (5.5)$$

де M – місячний посадовий оклад розробника;

T_p – кількість робочих днів у місяці, T_p = 22 дні; t – число днів роботи.

Розрахунки заробітної плати для розробників наведені в таблиці 5.6

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Керівник	25000	1136.36	7	7945.52
Розробник	20000	909.09	15	13648.5
Всього:				21594.02

Для робітників, які задіяні до даної розробки та працюють в наукових установах бюджетної форми за робочими професіями, заробітна плата розраховується за формулою:

$$З_Р = \sum_{i=1}^{n} t_i \cdot C_i \qquad (5.6)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i = \frac{M_M * K_i}{T_p * T_{ЗМ}} \quad (5.7)$$

де M_M – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., K_i – тарифний коефіцієнт робітника відповідного розряду, T_p = 22 дні; $T_{зм}$ – тривалість зміни = 8 годин.

Таблиця 5.7 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	56.88	1.54	455
Тестування	5	4	49.85	1.35	249.5
Впровадження	2	1	36.93	1	73.86
Всього					777.86

Потрібно розрахувати додаткову заробітну плату робітників. Вона розраховується як 10-12% від основної:

$$З_Д = (0,1...0,12) * З_О \quad (5.8)$$

$$З_Д = 0.1 * (З_О + З_Р) = 0,1 * (21594,02 + 777,86) = 2167,188 \text{ (грн.)}$$

Нарахування на заробітну плату $Н_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{ЗП} = (З_О + З_Р + З_Д) * \frac{\beta}{100} = (21594,02 + 777,86 + 2167,18) * 0,22 = 52599,78 \text{ (грн.)} \quad (5.9)$$

де $З_О$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата розробників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне страхування.

Розрахунок амортизаційних відрахувань виконується за такою формулою:

$$А = \frac{Ц \times Н_А}{100} * \frac{Т}{12} \quad (5.10)$$

де $Ц$ – балансова вартість обладнання, грн;

H_A – річна норма амортизаційних відрахувань. Для даного випадку можна прийняти, що H_A = 25%;

T – термін використання (T=2 місяців);

T_B – корисний час використання (T_B для комп'ютера становить 4 роки).

Під час виконання розробки методики тестування безпеки веб-застосунку використовувався ноутбук ціною в 25000 грн. Амортизаційні відрахування для нього представлені у таблиці 5.8.

Таблиця 5.8 – Амортизаційні відрахування

Найменування	Ціна, грн.	Норма амортизації, %	Термін використання, міс.	Сума амортизації, грн.
Ноутбук	25000	25	2	1041,06
Всього	1041,06			

Витрати на силову електроенергію розраховуються за формулою:

$$B_E = B*П*Ф*K_П \quad (5.11)$$

де B – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,74 кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$=176=(7+15)*8 год);

$K_П$ – коефіцієнт використання потужності ($K_П$ <1, $K_П$ = 0,8).

Відповідно до формули 5.11 витрати на силову електроенергію:

B_E=4,62*0,74*176*0,8=481,36 (грн.)

Інші витрати $B_{ін}$ можна прийняти як (100-300)% від суми основної заробітної плати розробників, які виконували роботу, тобто:

$$B_{ін}=1*(21594,02+777,86)=22371,88 \text{ (грн.)} \quad (5.11)$$

Сума усіх попередніх витрат дає загальні витрати на виконання роботи. Усі витрати складають:

В=21594,02+777,86+2167,18+481,36+5259,9+22371,88+1041,06 =52915,4 (грн.)

Розрахунок загальної вартості наукової розробки $В_{заг}$ за формулою:

$$B_{заг}=\frac{B}{\alpha}=52915,4 \quad (5.12)$$

де $\alpha=1$ – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відносних одиницях.

Прогнозування загальних витрат *ЗВ* на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ=\frac{В_{заг}}{\beta} \quad (5.13)$$

Розрахунок прогнозованих загальних витрат = 75593,42 (грн.)

5.2.2 Розрахунок собівартості розробки методики тестування безпеки для веб-застосунків

Витрати на силову електроенергію розраховуються за формулою:

$$В_Е=В*П*Ф*К_П \quad (5.14)$$

де $В$ – вартість 1кВт-години електроенергії (В=4,62 грн/кВт);

$П$ – установлена потужність комп'ютеру ($П$=0,74 кВт);

$Ф$ – фактична кількість годин роботи комп'ютеру ($Ф$=176=(7+15)*8 год);

$К_П$ – коефіцієнт використання потужності ($К_П$ <1, $К_П$ = 0,8).

Відповідно до формули 5.11 витрати на силову електроенергію:

$В_Е$=4,62*0,74*176*0,8=481,36 (грн.)

Для робітників, які задіяні до даної розробки та працюють в наукових установах бюджетної форми за робочими професіями, заробітна плата розраховується за формулою:

$$З_Р=\sum_{i=1}^{n} t_i \cdot C_i \quad (5.15)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин; n – число робіт по видах та розрядах; C_i – погодинна тарифна ставка робітника відповідного розряду, який виконує дану роботу. C_i визначається за формулою:

$$C_i=\frac{М_М*К_i}{Т_р*Т_{ЗМ}} \quad (5.16)$$

де $М_М$ – розмір мінімальної заробітної плати за місяць, грн.; в 2021 році мінімальна заробітна плата становить – 6500 грн., $К_i$ – тарифний коефіцієнт робітника відповідного розряду, $Т_р$ = 22 дні; $Т_{зм}$ – тривалість зміни = 8 годин.

Таблиця 5.9 – Заробітна плата робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Розробка	8	5	56.88	1.54	455
Тестування	5	4	49.85	1.35	249.5
Впровадження	2	1	36.93	1	73.86
Всього					777,86

Потрібно розрахувати додаткову заробітну плату робітників. Вона розраховується як 10-12% від основної:

$$З_Д=(0,1...0,12)*З_О \quad (5.17)$$

$$З_Д=0.1*(З_О+З_Р)=0,1*777,86=77,786 \text{ (грн.)}$$

Нарахування на заробітну плату $Н_{ЗП}$ розраховується як 22% від суми основної та додаткової заробітної плати:

$$Н_{ЗП}=(З_О+З_Р+З_Д)*\frac{\beta}{100}=(777,86+77,786)*0,22=188,24 \text{ (грн.)} \quad (5.18)$$

де $З_О$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_Д$ – додаткова заробітна плата розробників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне страхування.

Для розрахунку загальновиробничих витрат з рахунку на одиницю продукції буде використана наступна формула, яка відображає нормативи відносно до основної заробітної плати основних робітників, які виготовляють продукцію:

$$ЗВВ=Н_В*З_О \text{ (грн.)} \quad (5.19)$$

Норматив загальновиробничих витрат для програмних продуктів становить 230-270%.

ЗВВ=2,7*777,86=2100,22 (грн.)

Сума попередніх витрат утворює виробничу собівартість розробки.

$$S_В = 481,36+777,86+77,78+188,24+2100,22=3625,46 \text{ (грн.)} \quad (5.20)$$

5.3 Розрахунок мінімальної ціни та чистого прибутку від реалізації розробки методики тестування безпеки веб-застосунків

Ціна завжди залежить від виробництва та відображає рівень суспільних необхідних витрат. Тобто ціна це є грошовий показник вартості товару, продукції чи послуги.

Так як, будь-яка розробка зазвичай приймається та впроваджується лиза за завданням та вимогами замовника, або коли результатом розробки є якась продукція, що підпорядковується державному регулюванню, то нижню межу ціни саме реалізації розробки можна розрахувати за наступною формулою:

$$Ц=S_B*(1+\frac{P}{100})*(1+\frac{\omega}{100}) \quad (5.21)$$

де S_B – виробнича собівартість інноваційного рішення, грн.;

P – норматив рентабельності узгоджений із замовником або встановлений державою, (Р=30…60%);

ω – ставка податку на додану вартість, % (з осені 2021 року $\omega = 20\%$).

$$Ц=3625,43*(1+\frac{60}{100})*(1+\frac{20}{100})=6960,82 \text{ (грн.)} \quad (5.22)$$

Із врахуванням коефіцієнта якості ціна розробки становить 25755,03 грн.

Чистий прибуток від реалізації розробки можна розрахувати за формулою:

$$П=(Ц-\frac{(Ц-МР)\cdot f}{100}-S_B-\frac{q\cdot S_B}{100})*(1-\frac{h}{100})*РП \quad (5.23)$$

де $Ц$ – ціна розробки, грн.; $МР$ – вартість матеріальних та інших ресурсів, що були придбані виробником для виготовлення розробки ($МР$=(0,1…0,2) $Ц$), грн.; f – зустрічна ставка податку на додану вартість, %; S_B – виробнича собівартість розробки, грн.; q – норматив, який визначає величину адміністративних витрат, витрат на збут та інші операційні витрати, % (рекомендовано q=5…10%); h – ставка податку на прибуток, %, $РП$ – прогнозований попит продажів.

$$П=(25755,03-\frac{(25755,03-25755,03*0,2)\cdot 14}{100}-3625,46-\frac{5*3625,46}{100})*(1-\frac{18}{100})*2=31264,52 \quad (5.23)$$

Прогнозований чистий прибуток від реалізації розробки складає 31264,52 грн.

5.4 Розрахунок терміну окупності коштів вкладених у наукову розробку методики тестування безпеки для веб-застосунків

Термін окупності вкладених у реалізацію наукового проекту інвестицій розраховано за формулою 5.24:

$$T_{ОК}=\frac{ЗВ}{П}=\frac{75593,42}{31264,52}=2,42 \quad (5.24)$$

Оскільки $T_{ОК} < 3$ років, то фінансування наукової розробки методики тестування безпеки для веб-застосунків є доцільним.

5.5 Висновки до розділу

У даному розділі було здійснено обґрунтування економічної доцільності розробки, при цьому рівень комерційного потенціалу даної розробки є вище середнього. За результатами розрахунків було виявлено що абсолютний рівень якості методики тестування безпеки для веб-застосунків становить 3.7 бали, при цьому відносний рівень якості розробки становить 2.75.

Також було розраховано загальний показник конкурентоспроможності, який показав що дана методика є конкурентоспроможною, відображається в оцінці 3.5 бали. Загальні витрати для розробки методики, що стосуються виконавців розробки, склали 75593,42 грн, а собівартість розробки – 3625,46 грн.

Також було розраховано прогнозований чистий прибуток розробки, який склав 31264,52 грн, та мінімальну ціну розробки, яка становить 25755,03 грн. На основі отриманих даних було розраховано термін окупності вкладених інвестицій, який складає 2,42 роки, що є меншим значенням за 3, а тому фінансування розробки методики тестування безпеки веб-застосунків є доцільним.

ВИСНОВКИ

Аналіз роботи веб-застосунків показав необхідність перевірки безпеки веб-застосунка. Завдяки проведеному аналізу видив тестування, було зроблено висновок, що тестування безпеки є нечітко визначеним видом, а тому виникає плутанина в тому як тестувати саме безпеку та обґрунтовано необхідність розробки методики тестування безпеки як стану веб-застосунку, а не його функціональності. Серед розглянутих інструментів тестування було виділено саме ті інструменти, які можуть допомогти в тестування безпеки, оскільки вони надають більш розширені результати сканування веб-застосунків щодо вразливостей. На основі проведених аналізів інформаційних джерел, була виявлена необхідність у створенні методики тестування безпеки, яка перевірятиме безпеку веб-застосунку як один із його критеріїв якості.

Під час формування вимог до розробленої методики, було здійснено математичний опис моделі, що відображає процес розробки програмного забезпечення як сукупність множин. Було проаналізовано етапи життєвого циклу розробки програмного забезпечення для веб-застосунків та виділено основні критерії якості, яким мають відповідати сформовані вимоги. Вимоги були сформовані на основі результатів аналізу різних стандартів щодо забезпечення безпеки для інформаційно-комунікаційних систем, а основою для формування вимог став стандарт, який описує критерії якості безпеки для веб-застосунків. Після отриманих вимог було відібрано інструменти, що використовувалися в процесі перевірки коректності застосування розробленої методики тестування безпеки веб-застосунків.

Відповідно до вимог, був складений чек-ліст основних перевірок, який відображає сформовані раніше вимоги до розроблюваної методики. Кожен із пунктів чек-ліста має властивість декомпозиції, що дозволяє розробникам чи тестувальникам розбити кожну з перевірок на декілька інших, для написання кращих тест-кейсів та отримання кращого результату перевірок. На основі сформованого чек-ліста було розроблено низку тест-кейсів, які забезпечують

перевірку основних критеріїв якості, відібраної для розробки кейсів, вимоги. Кожен з цих тест-кейсів можна переробити та підлаштовувати під веб-застосунок для якого буде застосовуватися дана методика перевірки стану безпеки. Для аналізу отриманих результатів було розроблено очікувані результати, які відображають у відсотковому значення наскільки кожна з вимог має бути перевірена, щоб переконатися що стан безпеки перевірено якісно.

На основі розробок, було проведено перевірку коректності застосування методики тестування безпеки веб-застосунків. Тестування проводилося за допомогою розроблених тест-кейсів та допоміжних застосунків сканування безпеки. Отримані результати проаналізовані та було здійснено оцінку коректності застосування методики. Аналіз результатів показав, що перевірка стану безпеки за розроблюваною методикою була виконана коректно, а отримані звіти після сканування допоміжними застосунками відобразили доцільність розроблених вимог.

Оцінювання економічного потенціалу розробки методики тестування безпеки для веб-застосунків є доцільним, так як термін окупності становить менше трьох років. Оцінювання критеріїв якості доцільності економічної розробки показав, що розроблювана методика є більше ніж на половину ефективнішою за існуючі аналоги.

Таким чином, розроблена методика тестування безпеки веб-застосунків була сформована на основі аналізу інформаційних джерел, а вимоги до неї були сформовані на основі діючих стандартів з інформаційної безпеки. Реалізація та тестування даної методики показало, що її можна використовувати під час розробки та після отримання фінальної версії застосунку, що є гарним показником для якіснішого знаходження вразливостей застосунків. Розроблену методику можна використовувати для різного роду веб-застосунків, а особливо тих, де потрібно перевірити стан безпеки як один із критичних критеріїв якості.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a Web Application?: веб-сайт. URL: https://blog.stackpath.com/web-application/ (дата звернення: 02.09.2021)

2. ETSI TS 103 645 V1.1.1. Cyber: Cyber Security for Consumer Internet of Things. 2019. URL: https://www.etsi.org/deliver/etsi_ts/103600_103699/103645/01.01.01_60/ts_103645v010101p.pdf (дата звернення: 02.09.2021)

3. Куликов Святослав. Тестирование программного обеспечения. Базовый курс. 3-е издание. EPAM Systems, 2015р. 300 с.

4. Душко А. О., Баришев Ю. В. Засіб захищеного обміну даними на основі псевдонедетермінованих криптографічних перетворень. *XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету:* тези наук.-практ. конф. ВНТУ, Вінниця, 2020 р. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9327/7593

5. Душко А. О., Баришев Ю. В. Алгоритм потокового шифрування даних методом псевдонедетермінованих криптографічних перетворень. *Всеукраїнська науково-практична інтернет-конференція Молодь в науці: дослідження, проблеми, перспективи (МН-2021)* URL: https://conferences.vntu.edu.ua/index.php/mn/mn2021/paper/viewFile/11033/9189

6. Difference between Website and Web Application (Web App): веб-сайт. URL: https://www.guru99.com/difference-web-application-website.html (дата звернення: 04.09.2021)

7. Слободянюк О. В. Хаханова А. В., Комолов И. Д.Безпека інтернет ресурсів: аналіз розповсюдженості загроз та технології захисту. Системы и процессы управления: науч.-техн. журнал. 2018 р. № 2. 30-34 с.

8. What Is a Web Application? How It Works, Benefits and Examples: веб-сайт. URL: https://www.indeed.com/career-advice/career-development/what-is-web-application (дата звернення: 05.09.2021)

9. Web application security: веб-сайт. URL: https://digital.ai/glossary/web-application-security (дата звернення: 06.07.2021)

10. Що таке програмне забезпечення як послуга (SaaS)?: веб-сайт. URL: https://cbto.com.ua/library/saas (дата звернення: 10.09.2021)

11. Web application (Web app): веб-сайт. URL: https://searchsoftware quality.techtarget.com/definition/Web-application-Web-app (дата звернення: 12.09.2021)

12. What is web application security?: веб-сайт. URL: https://www.cloudflare.com/learning/security/what-is-web-application-secur ity/ (дата звернення: 12.09.2021)

13. Website security: веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security (дата звернення: 14.09.2021)

14. Kaner C., Falk J., Quoc Nguyen H. Testing computer software. Second edition. K: International Thomson Publishing Company, 2002 y. 538 p.

15. ISTQB. Advanced Security Tester Syllabus. Version 2018 V3.1. ISTQB: International Software Testing Qualifications Board, 2019 y. 93 p.

16. The different types of software testing: веб-сайт. URL: https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing (дата звернення: 18.09.2021)

17. Different Types of Testing in Software: 100 Examples: веб-сайт. URL: https://www.guru99.com/types-of-software-testing.html (дата звернення: 19.09.2021)

18. What is Security Testing?: веб-сайт. URL: https://www.guru99.com/what-is-security-testing.html (дата звернення: 21.09.2021)

19. Security Testing vs Penetration Test: веб-сайт. URL: https://dou.ua/lenta/columns/security-testing-vs-penetration-test/ (дата звернення: 23.09.2021)

20. Intruder: веб-сайт. URL: https://www.intruder.io/?utm_source=referral&utm_campaign=guru99_what_is_security_testing (дата звернення: 30.09.2021)

21. OWASP Dependency-Check: веб-сайт. URL: https://owasp.org/www-project-dependency-check/ (дата звернення: 02.10.2021)

22. OWASP ZAP: веб-сайт. URL: https://owasp.org/www-project-zap/ (дата звернення: 03.10.2021)

23. Acunetix: веб-сайт. URL: https://www.acunetix.com/plp/dast/?utm_medium=3rdparty&utm_source=guru99&utm_campaign=security-testing&utm_content=listing (дата звернення: 05.10.2021)

24. Wireshark: веб-сайт. URL: https://www.wireshark.org/ (дата звернення: 06.10.2021)

25. W3af: веб-сайт. URL: http://w3af.org/take-a-tour (дата звернення 06.10.2021)

26. System Development Life Cycle Guide: веб-сайт. URL: https://www.clouddefense.ai/blog/system-development-life-cycle (дата звернення: 12.10.2021)

27. A Comprehensive Guide To The 7 Phases of Web Development Life Cycle: веб-сайт. URL: https://www.monocubed.com/web-development-life-cycle/ (дата звернення: 13.10.2021)

28. Web Application Security Testing Guide: веб-сайт. URL: https://www.softwaretestinghelp.com/security-testing-of-web-applications/ (дата звернення: 15.10.2021)

29. Requirements Engineering for Web Applications - A Comparative Study: веб-сайт. URL: https://www.researchgate.net/publication/220538141_Requirements_Engineering_for_Web_Applications_-_A_Comparative_Study (дата звернення: 17.10.2021)

30. ISO 27001:2013. Информационные технологии – Методы защиты – Системы менеджмента информационной безопасности – Требования. [Чинний від 2013-10-01]. А. Горбунов. Переклад з англійської. 2013. 34 с.

31. NIST 800-53. Security and Privacy Controls for Federal Information Systems and Organizations. [Чинний від 2015-01-22]. National Institute of Standards and Technology: Patrick D. Gallagher. 2015. 172 p.

32. COBIT 5 Enabling Processes. 2012. URL: https://community. mis.temple.edu/mis5203sec951spring2021/files/2019/01/COBIT5-Ver2-enabli ng.pdf (дата звернення: 30.10.2021)

33. Who is the OWASP Foundation?: веб-сайт. URL: https://owasp.org/ (дата звернення: 01.11.2021)

34. OWASP ASVS 4.0.3. Application Security Verification Standard 4.0.3. [Чинний від 2021-10]. California: OWASP, Open Web Application Security Project, 2021. 71 p.

35. Test Documentation in Software Testing: веб-сайт. URL: https://www.guru99.com/testing-documentation.html (дата звернення: 05.11.2021)

36. How to Write Test Cases: Sample Template with Examples: веб-сайт. URL: https://www.guru99.com/test-case.html (дата звернення: 05.11.2021)

37. Дія. Державні послуги онлайн: веб-застосунок. URL: https://diia.gov.ua/ (дата звернення 01.12.2021)

38. ImmuniWeb: веб-сайт. URL: https://www.immuniweb.com/ (дата звернення: 02.12.2021)

ДОДАТКИ

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

ЗАТВЕРДЖУЮ
Завідувач кафедри ЗІ, д. т. н., проф.
_____ В. А. Лужецький
____ _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

на тему: «МЕТОДИКА ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ»
08-20.МКР.003.00.000 ТЗ

Керівник магістерської кваліфікаційної роботи
к. т. н., доц. каф. ЗІ Ю. В. Баришев

Вінниця 2021

1 Підстави для проведення робіт

Робота проводиться на підставі наказу ректора ВНТУ від 24 вересня 2021 року №277.

Дата початку роботи 01.09.2021 р.
Дата закінчення роботи 21.12.2021 р.

2 Мета та призначення МКР

Метою магістерської кваліфікаційної роботи є збільшення якості перевірки безпеки веб-застосунків за рахунок створення нової методики тестування безпеки, яка базується на діючих стандартах щодо забезпечення безпеки для інформаційно-комунікаційних систем, результатів дослідження можливих ризиків та загроз, а також розроблених чек-лістів та тест-кейсів.

Об'єктом дослідження є процес тестування безпеки веб-застосунків.

Предметом дослідження є методика тестування безпеки веб-застосунків.

Актуальність теми. На сьогоднішній день багато провідних компаній світу переводять свою роботу в онлайн режим. Навіть деякі державні послуги переводять в режим онлайн для зручності та безпечності користування ними людьми, тому постає питання щодо безпечності послуг онлайн. Відповідь на це питання є неоднозначною, адже з одного боку так, це буде безпечно, якщо розробники подбали про функціональні методи захисту, по типу шифрування чи автентифікації користувача. Тоді це буде перевірка безпеки як частини функціональності застосунку. Зазвичай, багато хто на цьому зупиняється і забуває про те, що існує чимала кількість способів зруйнувати або дістати інформацію, обійшовши функціональні методи захисту.

Звідси постає задача, як саме перевірити безпеку застосунку, але не як частину функціональності. Актуальним вирішенням даної проблеми може стати розробка методики тестування безпеки веб-застосунків, яка допоможе перевірити безпеку застосунку як один із його станів.

3 Вихідні дані для проведення МКР

МКР проводиться вперше і вхідними даними для проведення МКР є:

3.1 Куликов Святослав. Тестирование программного обеспечения. Базовый курс. 3-е издание. / Святослав Куликов – EPAM Systems, 2015р. – 300 с.

3.2 Слободянюк О. В. Безпека інтернет ресурсів: аналіз розповсюдженості загроз та технології захисту / О. В. Слободянюк, А. В. Хаханова, И. Д. Комолов // Системы и процессы управления: науч.-техн. журнал. – 2018 р. – № 2. – 30-34 с.

3.3 Баришев Ю. В. Дискреційна модель та метод розмежування прав доступу до розподілених інформаційних ресурсів / Ю. В. Баришев, В. А. Каплун, К. В. Неуйміна // Наукові праці ВНТУ. – №2, 2017. – 8 с.

3.4 Kaner C. Testing computer software. Second edition / C. Kaner, J. Falk, H. Quoc Nguyen. – K: International Thomson Publishing Company. – 2002 у. – 538 р.

3.5 ISTQB. Advanced Security Tester Syllabus. Version 2018 V3.1 / ISTQB: International Software Testing Qualifications Board. – 2019 у. – 93 р.

3.6 OWASP ASVS 4.0.3. Application Security Verification Standard 4.0.3. [Чинний від 2021-10]. California: OWASP, Open Web Application Security Project, 2021. 71 р.

4 Виконавці МКР

Студент групи 1 БС-20м Душко Аліна Олександрівна.

5 Вимоги до виконання МКР

Для збільшення якості перевірки безпеки веб-застосунків за рахунок створення нової методики тестування безпеки потрібно вирішити низку задач:

- проаналізувати відомі види та методики процесу тестування, стандарти NIST, ISO, COBIT, OWASP;
- проаналізувати можливі ризики та загрози;
- проаналізувати існуючі інструменти тестування безпеки;
- сформувати вимоги, розробити чек-лісти та тест-кейси для тестування безпеки веб-застосунків;
- сформувати очікувані результати методики;
- виконати перевірку коректності застосування методики.

6 Вимоги до супровідної документації

Графічна і текстова документація повинна відповідати діючим стандартам України.

7 Етапи МКР

Робота з теми виконується в 11 етапів

Зміст етапу	Початок	Закінчення	Очікувані результати	Звітна документація
Аналіз завдання. Вступ	01.09.2021	04.09.2021	Вступ	Чернетка вступу

Продовження таблиці 1

Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2021	15.09.2021	Перший розділ	Чернетка першого розділу
Науково-технічне обґрунтування	16.09.2021	22.09.2021	Аналіз існуючих аналогів. Вибір напрямку дослідження Аналіз відомих методів. Постановка завдання	Чернетка першого розділу
Розробка технічного завдання	23.09.2021	4.10.2021	Технічне завдання	Технічне завдання
Аналіз методик та методів тестування безпеки	05.10.2021	08.10.2021	Аналіз існуючих методик тестування	Чернетка першого розділу
Аналіз та формування вимог до методики	09.10.2021	16.10.2021	Аналіз стандартів Аналіз SDLC Формування вимог Формування моделі оцінки	Чернетка другого розділу
Розробка методики тестування безпеки веб-застосунків	17.10.2021	14.11.2021	Розроблені чек-лісти та тест-кейси Розроблені очікувані результати Розроблена методика	Чернетка третього розділу
Застосування методики тестування безпеки веб-застосунків	15.11.2021	17.11.2021	Проведення тестування методики	Чернетка третього розділу
Розробка розділу економічного обґрунтування доцільності розробки	18.11.2021	21.11.2021	Економічний розділ	Чернетка економічного розділу
Аналіз виконання ТЗ, висновки	22.11.2021	24.11.2021	Готові висновки	Чернетка висновків
Оформлення пояснювальної записки	25.11.2021	29.11.2021	Пояснювальна записка	Пояснювальна записка

8 Очікувані результати та порядок реалізації МКР

Передбачається розробка нової методики тестування, як б перевіряла безпеку як стан веб-застосунку, а не як частину її функціональності.

9 Матеріали які подаються після закінчення МКР

По завершенню роботи подається пояснювальна записка та ілюстративна частина.

10 Порядок приймання МКР та її етапів

Апробація на науково-технічних конференціях та семінарах. Результати роботи будуть розглядатися на засідання ДЕК із захисту магістерських кваліфікаційних робіт.

Попередній захист та доопрацювання МКР грудень 2021 р.

Представлення МКР до захисту 15 грудня 2021р.

Захист МКР 21-23 грудня 2021 р.

11 Вимоги до розроблення документації

Документація буде виконуватись за допомогою комп’ютерного набору у відповідності вимог ДСТУ 3008:2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання».

12 Вимоги щодо технічного захисту інформації з обмеженим доступом

У зв’язку з тим, що дана робота не містить інформації, що потребує захисту у відповідності до законів України, заходи з її технічного захисту не передбачаються.

Розробив студент групи 1БС-20м _____Душко А. О.

Додаток Б. Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	1	2	3	4	5
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	1	2	3	4	5
		наявних фахівців			
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Додаток В. Результат перевірки роботи на плагіат

Ім'я користувача:
Каплун В.А. ЗІ

ID перевірки:
1009715632

Дата перевірки:
19.12.2021 21:19:06 EET

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.12.2021 21:20:22 EET

ID користувача:
61408

Назва документа: **Душко (1)**

Кількість сторінок: **56** Кількість слів: **13567** Кількість символів: **98979** Розмір файлу: **1.43 MB** ID файлу: **1009714019**

0.21%
Схожість

Найбільша схожість: **0.21%** з джерелом з Бібліотеки (ID файлу: **1005731656**)

0.15% Джерела з Інтернету	2	Сторінка 58
0.21% Джерела з Бібліотеки	1	Сторінка 58

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0.02%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **15 слів** та **0%**)

0% Вилучення з Інтернету	40	Сторінка 59
0.02% Вилученого тексту з Бібліотеки	37	Сторінка 59

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	4

Схожість

Джерела з Інтернету 2

2	https://knowledge.allbest.ru/transport/3c0a65625b2ad79b4c53a89421216d27_0.html	0.15%
3	http://kntu.kr.ua/doc/zb_semik.doc	0.13%

Джерела з Бібліотеки 1

1	**1 - МКР Губницький 2020**	ID файлу: **1005731656**	Навчальний заклад: **Vinnytsia National Technical University**	0.21%

Вилучення

Вилучення 40

https://armgpublishing.sumdu.edu.ua/wp-content/uploads/mmi/volume-9-issue-1/mmi2018_1_184_195.pdf	33 джерела	0.1%
https://knowledge.allbest.ru/manufacture/2c0b65635b3bd68a4c43a89421306c37_0.html	3 джерела	0.07%
http://knuba.wcms.in.ua/wloads/22load.doc	2 джерела	0.07%
http://anokhina.vk.vntu.edu.ua/pub		0.06%
https://dspace.bdpu.org/bitstream/123456789/578/3/Koval%20L.%20V.%20Mystets%CA%B9ko-pedahohichnyy%20fenomen%20pi...		0.06%

Вилучення по Бібліотеці акаунту 37

ДП Кубишкін 2017	ID файлу: **2077082**	Навчальний заклад: **Vinnytsia National Technical University**	8 Джерело	0.08%
192КолібабаВВ2020МКР (1)	ID файлу: **1005687113**	Навчальний заклад: **Vinnytsia National Technical University**		0.07%
Марценюк_КІ19мсз	ID файлу: **1008130059**	Навчальний заклад: **Vinnytsia National Technical University**	24 Джерело	0.06%
_BDR_SnihurAK_131_1PM17b	ID файлу: **1008283600**	Навчальний заклад: **Vinnytsia National Technical University**		0.06%
122ДанченкоДБ2019	ID файлу: **1000770484**	Навчальний заклад: **Vinnytsia National Technical University**	3 Джерело	0.06%

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

РЕЗУЛЬТАТИ АНАЛІЗУ СТАНДАРТІВ

Аналіз стандарту ISO 27001:2013

Вимога	Задача/Ціль
1. Політика інформаційної безпеки	Забезпечення менеджменту та підтримки інформаційної безпеки у відповідності до вимог прийнятого бізнес напрямку та діючим законодавчим та нормативним вимогам
2. Організація інформаційної безпеки	Вплив на формування основних елементів управління для ініціювання та контролю впровадження та експлуатації засобів захисту інформації в організації
3. Безпека персоналу	Надання гарантії того, що працівники та люди, що працюють по контракту з компанією розуміють свої обов'язки та відповідають тим функціям та вимогам, які їм потрібно виконати
4. Класифікація інформації	Надання гарантії того, що інформація має рівень захисту, який відповідає її значущості в компанії
5. Використання носіїв інформації	Попередження несанкціонованого розкриття, зміни, переміщення чи знищення інформації, що зберігається на носії
6. Контроль доступу	Обмеження доступу інформації та носіям для її обробки, гарантування авторизованого доступу користувачам та попередження несанкціонованого доступу до систем та служб, відповідальність користувачів за їх інформації автентифікації, попередити несанкціонований доступ до систем та застосунків
7. Апаратне забезпечення	Попередження втрати, пошкодження, викрадення або компрометацію активів та порушення діяльності компанії
8. Безпека виробничої діяльності компанії	Гарантування відповідного використання та безпечного експлуатування засобів обробки інформації, гарантування того, що інформація та засоби обробки інформації захищені від непотрібного коду, забезпечити резервне копіювання даних, реєстрація усіх події, що були зроблені з кодом та системою, гарантування цілісності експлуатуючих систем та засобів, мінімізування впливу аудиту на експлуатуючі системи та засоби
9. Безпека обміну інформацією	Гарантування захисту інформації в мережах та на підтримуючих їх засобах обробки інформації, забезпечення безпеки інформації, що передається всередині компанії та за її межам
10. Придбання, розробка та обслуговування систем	Гарантування того, що інформаційна безпека є невід'ємною частиною інформаційних систем протягом всього їх життєвого циклу
11. Безпека в процесах розробки та підтримки систем	Ставить задачу гарантування, що міри щодо забезпечення інформаційної безпеки розроблені та реалізуються під час усього циклу розробки системи
12. Дані для тестування	Прописує правила того, що дані для тестування повинні ретельно відбиратися, повинні бути захищеними та контрольованими
13. Неперервність інформаційної безпеки	Неперервність забезпечення інформаційної безпеки має бути влаштована всередину системи менеджменту
14. Аналіз інформаційної безпеки	Гарантування того, що засоби забезпечення інформаційної безпеки влаштовані всередину системи та використовуються у відповідності до наявної політики безпеки компанії

					08-20.МКР.003.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати аналізу стандартів	Літ.	Арк. 1 / Аркушів 4
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТИ АНАЛІЗУ СТАНДАРТІВ

Аналіз стандарту NIST 800-53

Вимога	Задача/Ціль
1. Тестування плану на випадок непередбачених обставин / Взаємозв'язок з супутніми планами (CP-4(1))	Організація координує тестування плану на випадок надзвичайних ситуацій з організаційними елементами, відповідальними за дані плани
2. Відновлення та реконструювання інформаційної системи (CP-10)	Організація забезпечує відновлення та перебудову інформаційної системи до відомого стану після збою
3. Контроль фізичного доступу / Тестування на проникнення (PE-3(6))	Організація використовує процес тестування на проникнення, який включає, визначені організацією, неоголошені спроби обійти або зламати контроль безпеки, пов'язаний із фізичними точками доступу до об'єкта
4. Сканування вразливостей / Тестування на проникнення та аналіз (RA-5(9))	Організація проводить тестування на проникнення для визначених організацією інформаційних систем або компонентів систем
5. Тестування та оцінка безпеки розробником / Статичне тестування коду (SA-11(1))	Організація вимагає від розробника інформаційної системи, компонента системи або служби інформаційної системи використовувати інструменти статичного аналізу коду для виявлення загальних недоліків і документування результатів аналізу
6. Тестування та оцінка безпеки розробником / Тестування на проникнення (SA-11(5))	Організація вимагає, щоб розробник інформаційної системи, компонента системи або служби інформаційної системи виконував тестування на проникнення на визначеною організацією шириною/глибиною та з визначеними обмеженнями
7. Захист ланцюга передачі інформації / Тестування на проникнення (аналіз елементів, процесів та акторів) (SA-12(11))	Організація використовує аналіз, незалежний аналіз третьої сторони, організаційне незалежне тестування на проникнення щоб визначити вразливі елементи які пов'язані з інформаційною системою
8. Безпека розробки архітектури та дизайну / Структура тестування (SA-17(6))	Організація вимагає від розробника інформаційної системи, компонента системи або служби інформаційної системи структурувати апаратне, програмне та мікропрограмне забезпечення, що стосуються безпеки, для полегшення тестування
9. Захист від шкідливого коду / Тестування та верифікація (SI-3(6))	Тестує механізми захисту від шкідливого коду, вводячи в інформаційну систему відомий доброякісний, нерозповсюджуваний тестовий приклад; Перевіряє, що відбувається після виявлення тест-кейсом порушення та повідомляє про зв'язаний з ним інцидент
10. Моніторинг інформаційної системи / Засоби для моніторингу (SI-4(9))	Організація використовує інструменти для тестування, зокрема для моніторингу системи
11. Тестування, моніторинг та навчання системи (PM-14)	Переглядає плани тестування, навчання та моніторингу на відповідність стратегії управління ризиками організації та пріоритетам дій щодо реагування на ризики в масштабі всієї організації

					08-20.МКР.003.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати аналізу стандартів	Літ.	Арк. 2 / Аркушів 4
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТИ АНАЛІЗУ СТАНДАРТІВ

Аналіз стандарту COBIT 5 Enabling Processes

Вимога	Задача/Ціль
1. Підтримка активних елементів управління (APO01.03)	Створіть набір політик, щоб керувати очікуваннями щодо контролю системи щодо відповідних ключових тем, таких як якість, безпека, конфіденційність, внутрішній контроль
2. Підтримка відповідності політикам і процедурам (APO01.08)	Аналізуйте невідповідність та вживайте відповідних заходів (це може включати зміну вимог). Регулярно оцінюйте продуктивність системи та вживайте відповідних заходів (за потреби)
3. Аналіз ризиків (APO12.02)	Розробляйте корисну інформацію для підтримки рішень щодо ризику, які враховують важливість факторів ризику для бізнесу
4. Формулювання ризику (APO12.04)	Своєчасно надавати інформацію про поточний стан ризиків та можливостей, пов'язаних із системою, усім зацікавленим сторонам, необхідним для відповідного реагування
5. Визначте документ з набором дій з управління ризиками (APO12.05)	Керуйте можливостями зниження ризику до прийнятного рівня як вказано у документі
6. Відповідь на ризик (APO12.06)	Своєчасно реагуйте ефективними заходами, щоб обмежити масштаби втрат від подій, пов'язаних із системою
7. Створити та підтримувати систему управління інформаційною безпекою (СУІБ) (APO13.01)	Створення та підтримка СУІБ, яка забезпечує стандартний, формальний та безперервний підхід до управління безпекою інформації, забезпечуючи безпечні технології та бізнес-процеси, які відповідають вимогам бізнесу та управління безпекою підприємства
8. Визначте та керуйте планом попередження ризиків інформаційної безпеки (APO13.02)	Підтримуйте план інформаційної безпеки, який описує, як потрібно керувати ризиком інформаційної безпеки та узгоджувати його зі стратегією та архітектурою підприємства
9. Моніторинг і перевірка СУІБ (APO13.03)	Підтримуйте та регулярно повідомляйте про необхідність і переваги постійного покращення інформаційної безпеки. Збирайте та аналізуйте дані про СУІБ та покращуйте ефективність СУІБ. Виправляйте невідповідності, щоб запобігти повторенню
10. Управління програмним і проектним ризиком (BAI01.10)	Ризик, з яким стикається управління програмами та проектами, повинен бути встановлений та централізовано зареєстрований
11. Виконуйте контроль якості (BAI03.06)	Розробляйте, забезпечуйте ресурсами та виконуйте план забезпечення якості, узгоджений із СУЯ (система управління якістю), щоб отримати якість, зазначену у визначенні вимог та політиках та процедурах якості підприємства
12. Підготовка до тестування (BAI03.07)	Створіть план тестування та необхідні середовища для тестування окремих та інтегрованих компонентів рішення, включаючи бізнес-процеси та допоміжні послуги, програми та інфраструктуру
13. Виконання тестування (BAI03.08)	Постійно виконуйте тестування під час розробки, включаючи контрольне тестування, відповідно до визначеного плану тестування та практик розробки у відповідному середовищі. Визначайте, реєструйте та встановлюйте пріоритети помилок і проблем, виявлених під час тестування
14. План приймального тестування (BAI07.03)	Створіть план тестування на основі загальноприйнятих стандартів, які визначають ролі, відповідальність та критерії входу та виходу. Переконайтеся, що план схвалений відповідними сторонами
15. Створення середовища тестування (BAI07.04)	Визначте та встановіть безпечне тестове середовище, що відповідає запланованим бізнес-процесам та операційному середовищу системи, продуктивності та потужності, безпеці, внутрішньому контролю, операційній практиці, вимогам до якості даних та конфіденційності та робочих навантажень
16. Виконання приймального тестування (BAI07.05)	Тестування змінюється незалежно відповідно до визначеного плану тестування перед встановленням в робоче середовище
17. Керування безпекою мережі та підключення (DSS05.02)	Використовуйте заходи безпеки та відповідні процедури керування для захисту інформації до всіх підключень
18. Відстежування інфраструктури для подій, пов'язаних із безпекою (DSS05.07)	Використовуючи інструменти виявлення вторгнень, відстежуйте інфраструктуру на предмет несанкціонованого доступу та переконайтеся, що будь-які події інтегровані із загальним моніторингом подій та керуванням інцидентами
19. Контролювання обробки інформації (DSS06.02)	Виконувати діяльність бізнес-процесів і пов'язані з ними засоби контролю на основі ризиків підприємства, щоб гарантувати, що обробка інформації є дійсною, повною, точною, своєчасною та безпечною

					08-20.МКР.003.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати аналізу стандартів	Літ.	Арк. 3 / Аркушів 4
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТИ АНАЛІЗУ СТАНДАРТІВ

Аналіз стандарту OWASP ASVS 4.0.3

Вимога	Задача/Ціль
1. Життєвий цикл безпечної розробки програмного забезпечення (1.1.2)	Перевірте використання моделювання загроз для кожної зміни дизайну або планування спринту, щоб ідентифікувати загрози, спланувати контрзаходи, сприяти відповідним реагуванням на ризики та керувати тестуванням безпеки
2. Архітектура розробки, вхідних та вихідних потоків (1.1.5)	Перевірте використання безпечного життєвого циклу розробки програмного забезпечення, який стосується безпеки на всіх етапах розробки
3. Архітектура бізнес-логіки (1.11.1)	Перевірте доступність контрольного списку безпечного кодування, вимог безпеки, інструкцій або політики для всіх розробників і тестувальників
4. Архітектура безпечного завантаження файлів (1.12.2)	Запровадити відповідну політику безпеки вмісту (Content Security Policy), щоб зменшити ризик від векторів XSS або інших атак із завантаженого файлу
5. Архітектура конфігурації (1.14.1)	Перевірте відокремлення компонентів різних рівнів довіри за допомогою чітко визначених засобів керування безпекою, правил брандмауера, шлюзів API, зворотних проксі, хмарних груп безпеки або подібних механізмів
6. Безпека пароля (2.1.10)	Переконайтеся, що немає вимог до періодичної ротації облікових даних або історії паролів
7. Загальний дизайн контролю доступу (4.1.1)	Переконайтеся, що програма застосовує правила контролю доступу на рівні надійної служби, особливо якщо контроль доступу на стороні клієнта присутній і його можна обійти
8. Загальний дизайн контролю доступу (4.1.2)	Переконайтеся, що всі атрибути користувачів і даних, а також інформація про політику, які використовуються засобами контролю доступу, не можуть бути використані кінцевими користувачами без спеціального дозволу
9. Загальний дизайн контролю доступу (4.1.3)	Переконайтеся, що існує принцип найменших привілеїв – користувачі повинні мати доступ лише до функцій, файлів даних, URL-адрес, контролерів, служб та інших ресурсів, на які вони мають спеціальний дозвіл. Це передбачає захист від підробки та підвищення привілеїв
10. Запобігання розмежування даних (5.5.1)	Переконайтеся, що змінні об’єкти використовують перевірку цілісності або зашифровані, щоб запобігти створенню ворожих об’єктів або підробці даних.
11. Секретне управління (6.4.1)	Переконайтеся, що рішення для управління секретами, наприклад сховище ключів, використовується для безпечного створення, зберігання, контролю доступу та знищення секретів
12. Реєстрація конфіденційної інформації (7.1.1)	Переконайтеся, що програма не реєструє облікові дані чи деталі платежу. Маркери сеансу повинні зберігатися в журналах лише в незворотній хешованій формі
13. Реєстрація конфіденційної інформації (7.1.2)	Переконайтеся, що програма не реєструє інші конфіденційні дані, як визначено місцевими законами про конфіденційність або відповідною політикою безпеки
14. Реєстрація конфіденційної інформації (7.1.3)	Переконайтеся, що програма реєструє події, пов’язані з безпекою, включаючи успішні та невдалі події автентифікації, помилки контролю доступу, помилки декомпозиції та помилки перевірки введення
15. Регулярне логування (7.2.2)	Переконайтеся, що всі рішення щодо контролю доступу можна реєструвати, а всі невдалі рішення реєструються. Це має включати запити з відповідними метаданими, необхідними для розслідування безпеки
16. Захист журналу (7.3.3)	Переконайтеся, що журнали безпеки захищені від несанкціонованого доступу та модифікації
17. Загальний захист даних (8.1.1)	Перевірте, чи програма захищає конфіденційні дані від кешування в компонентах сервера, таких як балансувальники навантаження та кеші програм
18. Загальний захист даних (8.1.5)	Переконайтеся, що виконується регулярне резервне копіювання важливих даних і що виконується тестове відновлення даних
19. Загальний захист даних (8.1.6)	Переконайтеся, що резервні копії зберігаються безпечно, щоб запобігти крадіжці або пошкодженню даних

					08-20.МКР.003.00.000 І41		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати аналізу стандартів	Літ.	Арк. 4 / Аркушів 4
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТ ФОРМУВАННЯ ВИМОГ ДО МЕТОДИКИ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-ЗАСТОСУНКІВ

№	Вимога	Етап SDLC								Стандарт			
		1	2	3	4	5	6	7	8	ISO	NIST	COBIT	OWASP
1	Resource Planning	+	+							+		+	+
2	Requirements Analysis	+	+	+	+		+	+	+	+	+	+	+
3	SDLC/STLC Review		+	+		+	+	+	+	+		+	+
4	Design		+	+	+	+	+			+	+		+
5	Privacy Policy Analysis		+		+		+	+	+	+		+	+
6	Coding and Unit Testing					+	+				+		+
7	System Testing					+	+	+			+		+
8	Environment Review		+	+		+	+	+		+	+	+	+
9	Securing Files/Databases			+	+		+	+		+	+	+	+
10	WEB-application Testing					+	+	+	+	+			+

					08-20.МКР.003.00.000 І42		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результат формування вимог до методики тестування безпеки веб-застосунків	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

ПРИКЛАД ГОТОВОГО ЧЕК-ЛІСТА

№	Перевірка	Environment		
		Google Chrome	Opera	Safari
S1	Resource Planning			
S2	Requirements Analysis			
S3	SDLC/STLC Review			
S4	Design			
S5	Privacy Policy Analysis			
S6	Coding and Unit Testing			
S7	System Testing			
S8	Environment Review			
S9	Securing Files/Databases			
S10	WEB-application Testing			

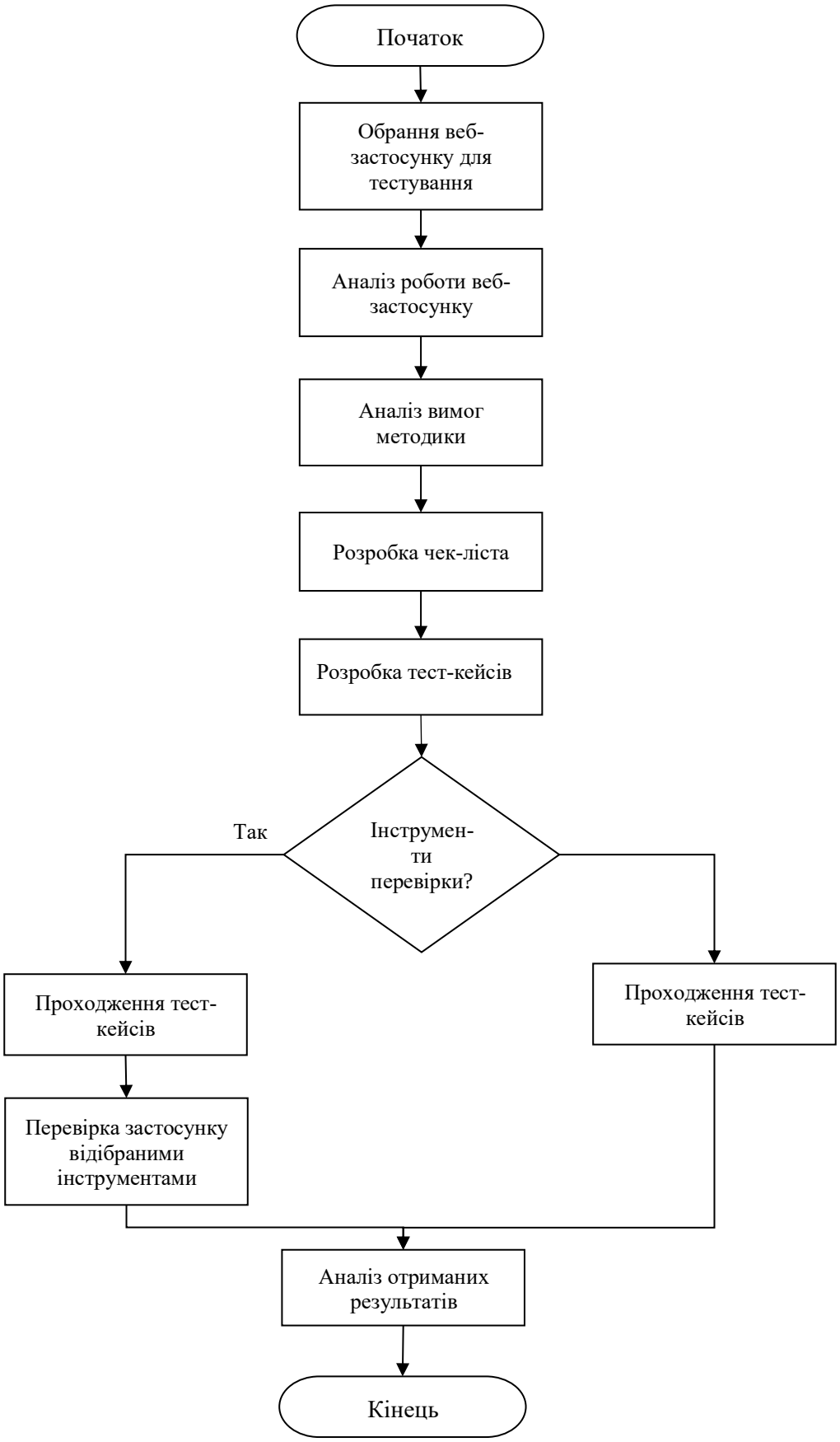
					08-20.МКР.003.00.000 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Приклад готового чек-ліста	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

ПРИКЛАД ГОТОВИХ ТЕСТ-КЕЙСІВ

№	Пункт чек-ліста	Тест-кейс	Статус
1	S10	Перевірити відображення усієї інформації веб-застосунку	
2	S10	Перевірити конфігурацію доступів веб-застосунку	
3	S10	Перевірити чи мережа та комп'ютер використовують брандмауер	
4	S10	Перевірити, що конфіденційні дані не передаються через URL	
5	S10	Перевірити через маніпулювання URL адресою, що застосунок не відображає небажану інформацію	
6	S10	Перевірити, що паролі зашифровані та передаються безпечно, наприклад, через HTTPS	
7	S10	Перевірити інформацію, яка зберігається в куках та кеші (повинна мати читабельний вигляд)	
8	S10	Перевірити, що вся конфіденційна інформація користувача та застосунку замаскована	
9	S10	Перевірити, що веб-застосунок жорстко не запрограмований на ім'я користувача чи його пароль	
10	S10	Перевірити, що введений пароль відповідає стандартним вимогам (8 length, великі/малі літери, цифри, символи)	
11	S10	Перевірити, що функція скидання паролю є безпечною	
12	S10	Перевірити, що персональні дані користувача не використовуються в нормальному вигляді	
13	S10	Перевірити, що функція запам'ятовування даних не використовує авто-заповнення доля паролів та фінансових даних	
14	S10	Перевірити, що веб-застосунок захищений щодо несанкціонованого доступу	
15	S10	Перевірити, що застосунок дозволяє завантажувати файли, перевіряючи їх при цьому через антивірус	
16	S10	Перевірити, що застосунок не використовує відкритий порт	
17	S10	Перевірити конфігурацію мережі (Wi-Fi)	
18	S10	Перевірити запити HTTPS, які використовує застосунок (PUT/DELETE запити заборонені до використання)	
19	S10	Перевірити, що сторінка логіну та паролю має містити інформативні повідомлення про помилки	
20	S10	Перевірити, що збій в роботі веб-застосунку не повинен викликати відображення конфіденційної інформації	
21	S10	Перевірити, що деталі внутрішньої системи не мають бути ні в жодному з повідомлень про помилки	
22	S10	Перевірити записи логування (конфіденційна інформація не повинна бути там відображена)	
23	S10	Перевірити, що журнали логування подій ведуться з відповідними доступами	
24	S10	Перевірити, що критичні ресурси веб-застосунку мають бути доступні тільки авторизованим користувачам та службам	
25	S10	Перевірити, що сеанс користувача закінчується після виходу чи закриття веб-застосунку	
26	S10	Перевірити, що використовується актуальна версія веб-застосунку	
27	S10	Перевірити, що веб-застосунок захищений до атак формату введення даних	
28	S10	Перевірити, що веб-застосунок захищений до атак типу грубої сили	

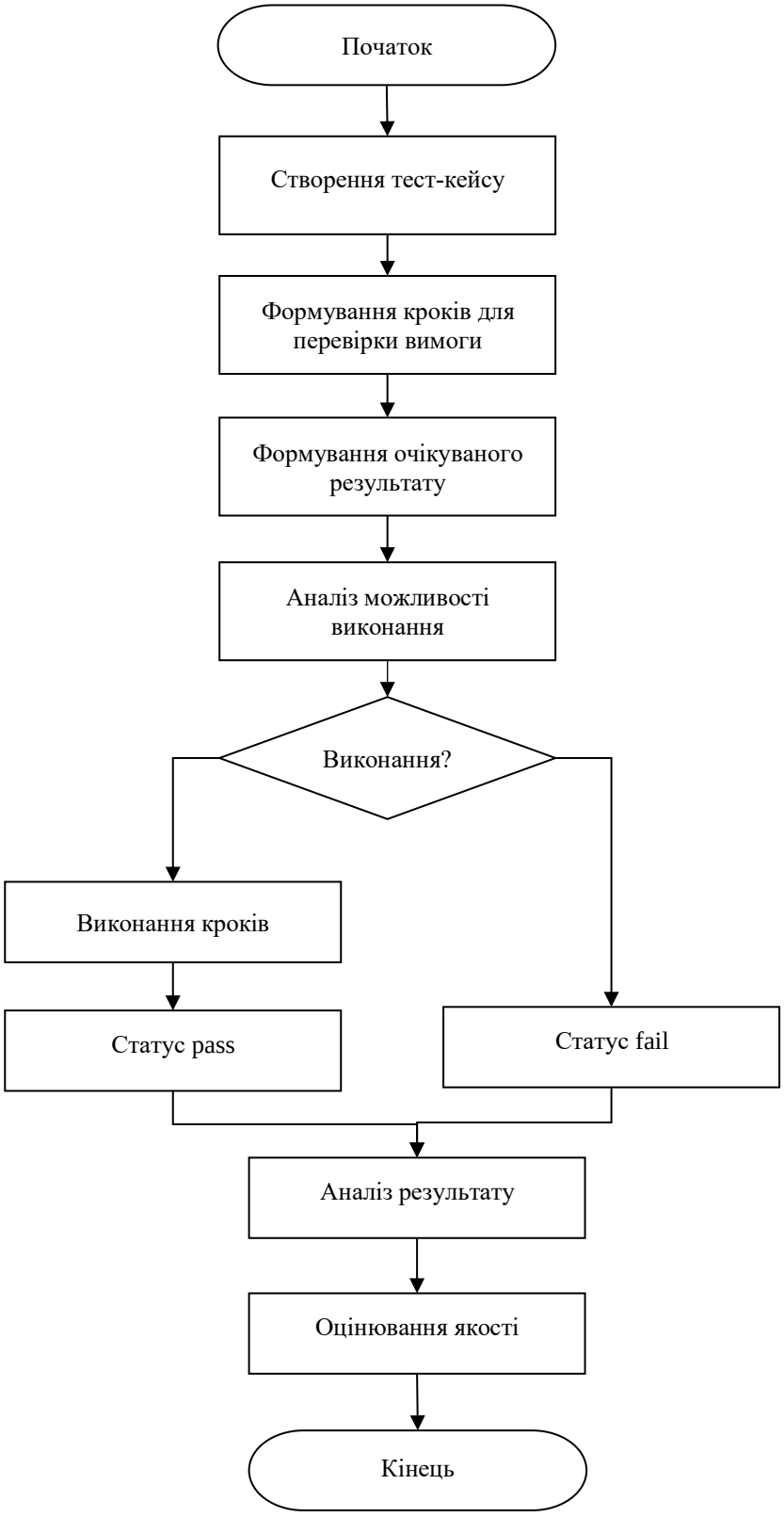
					08-20.МКР.003.00.000 І44			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Приклад готових тест-кейсів	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					1	1
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						

АЛГОРИТМ ЗАСТОСУВАННЯ МЕТОДИКИ



					08-20.МКР.003.00.000 145		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Алгоритм застосування методики	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

АЛГОРИТМ ПРОВЕДЕННЯ ПЕРЕВІРКИ БЕЗПЕКИ ЗА ВІДІБРАНИМ ТЕСТ-КЕЙСОМ



					08-20.МКР.003.00.000 І46		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Алгоритм проведення перевірки безпеки за відібраним тест-кейсом	Літ.	Арк. 1 / Аркушів 1
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

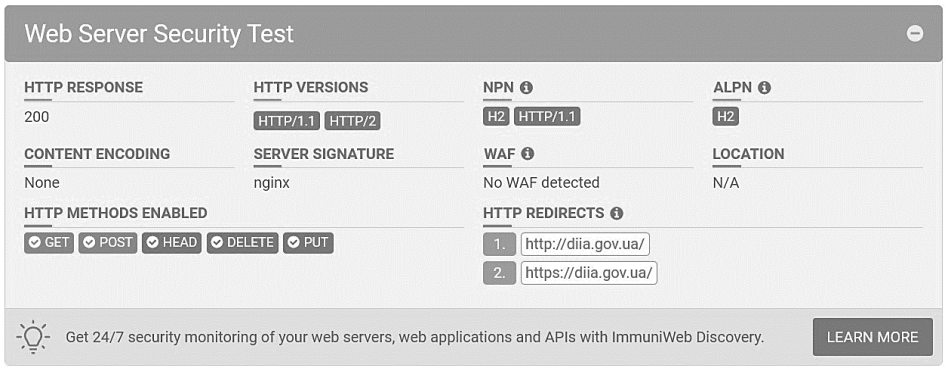
РЕЗУЛЬТАТИ ТЕСТУВАННЯ

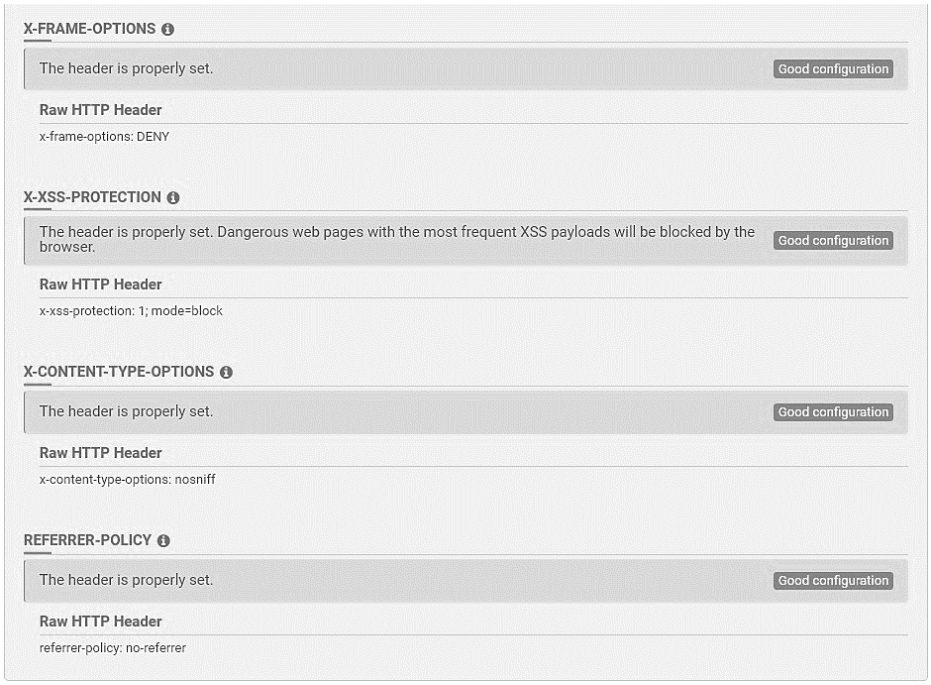
№	Пункт чек-ліста	Тест-кейс	Статус
1	S10	Перевірити відображення усієї інформації веб-застосунку	Pass
2	S10	Перевірити конфігурацію доступів веб-застосунку	Fail
3	S10	Перевірити чи мережа та комп’ютер використовують брандмауер	Pass
4	S10	Перевірити, що конфіденційні дані не передаються через URL	Pass
5	S10	Перевірити через маніпулювання URL адресою, що застосунок не відображає небажану інформацію	Pass
6	S10	Перевірити, що паролі зашифровані та передаються безпечно, наприклад, через HTTPS	Pass
7	S10	Перевірити інформацію, яка зберігається в куках та кеші (повинна мати читабельний вигляд)	Pass
8	S10	Перевірити, що вся конфіденційна інформація користувача та застосунку замаскована	Pass
9	S10	Перевірити, що веб-застосунок жорстко не запрограмований на ім’я користувача чи його пароль	Fail
10	S10	Перевірити, що введений пароль відповідає стандартним вимогам (8 length, великі/малі літери, цифри, символи)	Fail
11	S10	Перевірити, що функція скидання паролю є безпечною	Fail
12	S10	Перевірити, що персональні дані користувача не використовуються в нормальному вигляді	Pass
13	S10	Перевірити, що функція запам’ятовування даних не використовує авто-заповнення доля паролів та фінансових даних	Pass
14	S10	Перевірити, що веб-застосунок захищений щодо несанкціонованого доступу	Pass
15	S10	Перевірити, що застосунок дозволяє завантажувати файли, перевіряючи їх при цьому через антивірус	Fail
16	S10	Перевірити, що застосунок не використовує відкритий порт	Pass
17	S10	Перевірити конфігурацію мережі (Wi-Fi)	Pass
18	S10	Перевірити запити HTTPS, які використовує застосунок (PUT/DELETE запити заборонені до використання)	Pass
19	S10	Перевірити, що сторінка логіну та паролю має містити інформативні повідомлення про помилки	Pass
20	S10	Перевірити, що збій в роботі веб-застосунку не повинен викликати відображення конфіденційної інформації	Pass
21	S10	Перевірити, що деталі внутрішньої системи не мають бути ні в жодному з повідомлень про помилки	Pass
22	S10	Перевірити записи логування (конфіденційна інформація не повинна бути там відображена)	Pass
23	S10	Перевірити, що журнали логування подій ведуться з відповідними доступами	Fail
24	S10	Перевірити, що критичні ресурси веб-застосунку мають бути доступні тільки авторизованим користувачам та службам	Pass
25	S10	Перевірити, що сеанс користувача закінчується після виходу чи закриття веб-застосунку	Pass
26	S10	Перевірити, що використовується актуальна версія веб-застосунку	Pass
27	S10	Перевірити, що веб-застосунок захищений до атак формату введення даних	Pass
28	S10	Перевірити, що веб-застосунок захищений до атак типу грубої сили	Pass

					08-20.МКР.003.00.000 І47		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати тестування	Літ.	Арк. 1 / Аркушів 5
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Результати тестування за допомогою ImmuniWeb





					08-20.МКР.003.00.000 І47
Змн.	Арк.	№ докум.	Підпис	Дата	
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати тестування
Перевір.		Баришев Ю. В.			
Реценз.		Савицька Л. А.			
Н. Контр.		Баришев Ю. В.			
Затверд.		Лужецький В.А.			

Літ.	Арк.	Аркушів
	2	5

ВНТУ, 1БС-20м

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Результати тестування за допомогою ImmuniWeb

Content Security Policy Test

CONTENT-SECURITY-POLICY

Some directives have values that are too permissive. — Misconfiguration or weakness

The header contains deprecated directive(s). — Misconfiguration or weakness

Content-Security Policy is enforced. — Good configuration

Raw HTTP Header

Content-Security-Policy: default-src https: 'unsafe-inline' data:; script-src https: 'unsafe-eval' 'unsafe-inline' https://www.googletagmanager.com; style-src https: 'unsafe-inline'; object-src 'none'; report-uri /csp-hotline.php

Web Software Security Test

Get 24/7 security monitoring of your web applications and APIs with ImmuniWeb Discovery. LEARN MORE

Web Software Found	Web Software Outdated	Web Software Vulnerabilities
4	4	2

FINGERPRINTED CMS & VULNERABILITIES

No CMS were fingerprinted on the website. — Information

FINGERPRINTED CMS COMPONENTS & VULNERABILITIES

jQuery 3.4.0

The fingerprinted component version is outdated and vulnerable to publicly known vulnerabilities. Urgently update to the most recent version **3.6.0**.

CVSSv3.1 Score	Vulnerability CVE-ID	Vulnerability Type
5.5 Medium	CVE-2020-11022	CWE-79 — Cross-site scripting
4.2 Medium	CVE-2020-11023	CWE-79 — Cross-site scripting

Bootstrap 4.3.1

The component is outdated. No known security vulnerabilities found. Update to the most recent version **5.1.3**.

Cookies Privacy and Security Analysis

No cookies were sent by the web application. — Good configuration

External Content Privacy and Security Analysis

EXTERNAL CONTENT ON HOMEPAGE

External web content (e.g. images, video, CSS or JavaScript) can improve website loading time. However, the external content can also put privacy of website visitors at risk given that some information about them is transmitted to the third parties operating the external resources, sometimes even without proper HTTPS encryption or user consent.

External HTTP Requests	Failed HTTP Requests
11	0

cdn.ravenjs.com

https://cdn.ravenjs.com/3.26.4/raven.min.js DETAILS

www.googletagmanager.com

https://www.googletagmanager.com/gtm.js?id=GTM-WBX3V3Z DETAILS

connect.facebook.net

https://connect.facebook.net/en_US/fbevents.js DETAILS

https://connect.facebook.net/signals/config/2853763384889398?v=2.9.48&r=stable DETAILS

www.google-analytics.com

https://www.google-analytics.com/analytics.js DETAILS

					08-20.МКР.003.00.000 147		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати тестування	Літ.	Арк. 3 / Аркушів 5
Перевір.		Баришев Ю. В.					
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м	
Н. Контр.		Баришев Ю. В.					
Затверд.		Лужецький В.А.					

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Результати тестування за допомогою Acunetix

					08-20.МКР.003.00.000 І47			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати тестування	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					4	5
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Результати тестування за допомогою Acunetix

Error message on page

Medium Open

Vulnerability description

This alert requires manual confirmation

Application error or warning messages may expose sensitive information about an application's internal workings to an attacker.

Acunetix found an error or warning message that may disclose sensitive information. The message may also contain the location of the file that produced an unhandled exception. Consult the 'Attack details' section for more information about the affected page.

The vulnerability affects **https://diia.gov.ua/**

Discovered by **Scripting (Text_Search_File.script)**

Attack details

Not available in the free trial

HTTP request

The impact of this vulnerability

Error messages may disclose sensitive information which can be used to escalate attacks.

How to fix this vulnerability

Verify that this page is disclosing error or warning messages and properly configure the application to log errors to a file instead of displaying the error to the user.

Classification

CWE CWE-200
CVSS Base score: **5.3** — CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Attack Vector: Network
Attack Complexity: Low
Privileges Required: None
User Interaction: None
Scope: Unchanged
Confidentiality: Low
Integrity: None
Availability: None

Detailed information

Web References

- PHP Runtime Configuration
- Improper Error Handling

HTML form without CSRF protection

Medium Open

Vulnerability description

This alert requires manual confirmation

Cross-Site Request Forgery (CSRF, or XSRF) is a vulnerability wherein an attacker tricks a victim into making a request the victim did not intend to make. Therefore, with CSRF, an attacker abuses the trust a web application has with a victim's browser.

Acunetix found an HTML form with no apparent anti-CSRF protection implemented. Consult the 'Attack details' section for more information about the affected HTML form.

The vulnerability affects **https://diia.gov.ua/**

Discovered by **Crawler**

Attack details

Not available in the free trial

HTTP request

The impact of this vulnerability

An attacker could use CSRF to trick a victim into accessing a website hosted by the attacker, or clicking a URL containing malicious or unauthorized requests.

CSRF is a type of 'confused deputy' attack which leverages the authentication and authorization of the victim when the forged request is being sent to the web server. Therefore, if a CSRF vulnerability could affect highly privileged users such as administrators full application compromise may be possible.

How to fix this vulnerability

Verify if this form requires anti-CSRF protection and implement CSRF countermeasures if necessary.

The recommended and the most widely used technique for preventing CSRF attacks is know as an anti-CSRF token, also sometimes referred to as a synchronizer token. The characteristics of a well designed anti-CSRF system involve the following attributes.

- The anti-CSRF token should be unique for each user session
- The session should automatically expire after a suitable amount of time
- The anti-CSRF token should be a cryptographically random value of significant length
- The anti-CSRF token should be cryptographically secure, that is, generated by a strong Pseudo-Random Number Generator (PRNG) algorithm
- The anti-CSRF token is added as a hidden field for forms, or within URLs (only necessary if GET requests cause state changes, that is, GET requests are not idempotent)
- The server should reject the requested action if the anti-CSRF token fails validation

When a user submits a form or makes some other authenticated request that requires a Cookie, the anti-CSRF token should be included in the request. Then, the web application will then verify the existence and correctness of this token before processing the request. If the token is missing or incorrect, the request can be rejected.

Cookie(s) without Secure flag set

Low Open

Vulnerability description

This cookie does not have the Secure flag set. When a cookie is set with the Secure flag, it instructs the browser that the cookie can only be accessed over secure SSL channels. This is an important security protection for session cookies.

The vulnerability affects **https://diia.gov.ua/**

Discovered by **Crawler**

Attack details

Not available in the free trial

HTTP request

The impact of this vulnerability

None

How to fix this vulnerability

If possible, you should set the Secure flag for this cookie.

Classification

CWE CWE-16
CVSS Base score: **0** — AV:N/AC:L/Au:N/C:N/I:N/A:N
Access Vector: Network
Access Complexity: Low
Authentication: None
Confidentiality Impact: None
Integrity Impact: None
Availability Impact: None

					08-20.МКР.003.00.000 І47			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб..		Душко А. О.			Методика тестування безпеки веб-застосунків. Результати тестування	Літ.	Арк.	Аркушів
Перевір.		Баришев Ю. В.					5	5
Реценз.		Савицька Л. А.				ВНТУ, 1БС-20м		
Н. Контр.		Баришев Ю. В.						
Затверд.		Лужецький В.А.						