МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ
КАФЕДРА УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ ТА КІБЕРНЕТИЧНОЮ БЕЗПЕКОЮ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: “ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ, АНАЛІЗУ ТА ОЦІНЮВАННЯ ВРАЗЛИВОСТЕЙ СУЧАСНИХ ВЕБ-ДОДАТКІВ”

на здобуття освітнього ступеня магістра
зі спеціальності 125 Кібербезпека
освітньо-професійної програми Управління інформаційною безпекою

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ярослав УХАНЬ
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав: Здобувач вищої освіти гр. УБДМ 61
Ярослав УХАНЬ

Керівник:
к.т.н. Дмитро РАБЧУН

Рецензент:
к.т.н., доцент Юрій ПЕПА

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ
ТЕХНОЛОГІЙ

Навчально-науковий інститут захисту інформації

Кафедра Управління інформаційною та кібернетичною безпекою

Ступінь вищої освіти магістр

Спеціальність 125 Кібербезпека
Освітньо-професійна програма Управління інформаційною безпекою

ЗАТВЕРДЖУЮ
Завідувач кафедрою УІКБ
_____ Світлана ЛЕГОМІНОВА
"____"_____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Уханю Ярославу Валерійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: "Технології виявлення, аналізу та оцінювання вразливостей веб-додатків"

керівник кваліфікаційної роботи Дмитро РАБЧУН, к.т.н.
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від "19" жовтня 2023 р. №145

2. Строк подання кваліфікаційної роботи "18" грудня 2023 р

3. Вихідні дані до кваліфікаційної роботи: *інформаційна безпека, інтернет-технології, веб-додатки, методи та засоби оцінювання, аналіз вразливостей, стандарти, міжнародні стандарти, наукова та технічна література*

4. Перелік питань, які потрібно розробити:
1. Здійснити аналіз наукової літератури та публікацій, пов'язаних із виявленням, аналізом та оцінюванням вразливостей веб-додатків.
2. Дослідити сучасний стан вирішення проблеми захисту веб-додатків та зробити аналіз методів виявлення та оцінювання вразливостей веб-додатків.
3. На основі аналізу розробити алгоритми та комплексну методику, яка включає виявлення та оцінювання вразливостей сучасних веб-додатків з використанням штучного інтелекту.
4. Здійснити моніторинг та оцінку ефективності розробленої методики та відпрацювати пропозиції з практичного застосування.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання "02" жовтня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Визначення об’єкту, предмету, мети та завдань дослідження.	10.10.2023	
2.	Збір та аналіз літератури.	23.10.2023	
3.	Аналіз основних понять, характеристик та структури веб-додатків.	27.10.2023	
4.	Дослідження вимог міжнародних стандартів щодо захисту веб-додатків	10.11.2023	
5.	Аналіз технологій виявлення та оцінювання вразливостей веб-додатків.	15.11.2023	
6.	Розробка комплексної методики виявлення, аналізу та оцінювання вразливостей веб-додатків	22.11.2023	
7.	Відпрацювання пропозицій та рекомендацій удосконалення веб-додатків	29.11.2023	
8.	Оформлення роботи.	04.12.2023	
9.	Оформлення презентації.	14.12.2023	
10.	Отримання рецензії на роботу.	18.12.2023	
11.	Захист в ДЕК.	.01.2024	

Здобувачка вищої освіти _____ Ярослав УХАНЬ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Дмитро РАБЧУН
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЗАХИСТУ ІНФОРМАЦІЇ

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Ухань Я.В. до захисту кваліфікаційної роботи
(прізвище та ініціали)

за спеціальністю 125 Кібербезпека
(код, найменування спеціальності)

Освітньо-професійної програми Управління інформаційною безпекою
(назва)

на тему: “ Технології виявлення, аналізу та оцінювання вразливостей веб-додатків ”
Кваліфікаційна робота і рецензія додаються.

Директор ННІЗІ _____ Віталій САВЧЕНКО
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач **УХАНЬ Ярослав** у кваліфікаційній роботі дослідив сучасний стан захисту веб-додатків. Проаналізовані теоретичні аспекти безпеки веб-додатків дали змогу встановити нагальну потребу у аналізі технологій виявлення та оцінювання вразливостей у зв’язку з швидким розвитком сучасних інформаційних технологій. Встановлено, що одним із шляхів підвищення ефективності веб-безпеки є розроблення комплексних методик виявлення, аналізу та оцінювання вразливостей веб-додатків на застосуванні штучного інтелекту, нейронних мереж, машинного навчання. Розроблена комплексна методика виявлення аналізу та оцінювання вразливостей та відпрацьовані пропозиції по її застосуванню. **УХАНЬ Ярослав** показав розуміння проблеми дослідження та бачення основних теоретичних та практичних напрямів її вирішення, довів уміння самостійного застосування методів наукового дослідження, проявив себе як організований, відповідальний виконавець. Результати дослідження апробовані на науково-практичній конференції.
Це дозволяє оцінити виконану кваліфікаційну роботу здобувача **УХАНЯ Ярослава** на оцінку “відмінно” та присвоїти йому кваліфікацію “Магістр кібербезпеки за освітньо-професійною програмою “Управління інформаційною безпекою””.

Керівник кваліфікаційної роботи _____ Дмитро РАБЧУН
(підпис) *(Ім’я, ПРІЗВИЩЕ)*
“____“_____2024 року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута . Здобувач Ухань Я.В. допускається до захисту роботи в екзаменаційній комісії.

Завідувач кафедрою
Управління інформаційною
та кібернетичною безпекою _____ Світлана ЛЕГОМІНОВА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну магістерську роботу

здобувача вищої освіти Уханя Ярослава Валерійовича
на тему “ ТЕХНОЛОГІЇ ВИЯВЛЕННЯ, АНАЛІЗУ ТА ОЦІНЮВАННЯ ВРАЗЛИВОСТЕЙ ВЕБ-ДОДАТКІВ”

Актуальність. В сучасному цифровому світі різними веб-додатками користуються всі від окремих людей до великих компаній та державних організацій. Кіберзлочинці використовують такі веб-додатки для здійснення кібератак на об’єкти критичної інфраструктури, захист яких є надзвичайно важливим у сучасних умовах і потребує проведення наукових досліджень та пошуку новітніх способів захисту. Тому тема дослідження є надзвичайно актуальною.

Позитивні сторони. В роботі на основі проведеного аналізу сучасних методів розроблена комплексна методика, яка враховує інструменти виявлення, аналізу та оцінювання вразливостей веб-додатків в комплексі з урахуванням штучного інтелекту. За результатами дослідження запропоновані рекомендації постачальникам щодо оброблення вразливостей веб-додатків та обміну досвідом із застосуванням штучного інтелекту. Удосконалені критерії оцінювання систем захисту дають змогу встановити у відповідності із міжнародними стандартами ефективність розроблених стратегій веб-безпеки додатків врахуванням всіх вимог безпеки. Надані конкретні пропозиції та рекомендації щодо практичного впровадження розробленої комплексної методики в реальному середовищі.
Практичне впровадження розробленої методики може значно підвищити рівень безпеки веб-додатків, сприяючи уникненню атак та зменшенню ризиків для користувачів та організацій. Результати даної роботи можуть слугувати підґрунтям для подальших досліджень та вдосконалення стратегій захисту веб-додатків у змінних умовах кіберзагроз.

Недоліки. Разом з тим, доцільно було б застосувати при проведенні дослідження і інші бібліотеки Python для порівняння результатів, але це не зменшує важливості даної роботи та її науковості.

Висновок: кваліфікаційна робота виконана на належному науково-методичному рівні і заслуговує оцінки «відмінно» а її автор Ухань Ярослав Валерійович - присвоєння кваліфікації «Магістр кібербезпеки» за освітньо-професійною програмою «Управління інформаційною безпекою».

Рецензент: завідувач кафедри
Робототехніки та технічних систем
к.т.н., доцент

Юрій ПЕПА

_____
підпис

(Ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 66 стор., 27 рис., 63 джерела.

Метою роботи є визначення та розробка нових технологій для ефективного виявлення, аналізу та оцінювання вразливостей веб-додатків з метою підвищення рівня їхньої безпеки.

Об'єктом дослідження є технології виявлення, аналізу та оцінювання вразливостей веб-додатків.

Предмет дослідження – ефективність технологій виявлення, аналізу та оцінювання вразливостей веб-додатків.

Методи дослідження. Для вирішення завдань застосовані методи дослідження: аналіз і синтез при дослідженні наукових робіт, присвячених організації безпеки веб-додатків; моделювання при розробленні комплексної методики виявлення, аналізу та оцінюванні вразливостей сучасних веб-додатків.

Короткий зміст роботи. Проаналізовані теоретичні аспекти безпеки веб-додатків, визначена їх характеристика та структура і вимоги міжнародних стандартів щодо захисту дали змогу встановити нагальну потребу у аналізі технологій виявлення та оцінювання вразливостей у зв'язку з швидким розвитком сучасних інформаційних технологій.

Виявлено, що сучасні методи виявлення вразливостей не завжди задовольняють вимогам ефективної безпеки веб-ресурсів та потребують пошуку нових підходів до організації інформаційної безпеки веб-ресурсів.

Встановлено, що одним із шляхів підвищення ефективності веб-безпеки є розроблення комплексних методик виявлення, аналізу та оцінювання вразливостей веб-додатків, які повинні ґрунтуватись на застосуванні штучного інтелекту, нейронних мереж, машинного навчання.

Удосконалені критерії оцінювання дають змогу встановити у відповідності із міжнародними стандартами ефективність розроблених стратегій веб-безпеки додатків врахуванням всіх вимог безпеки.

На основі проведеного аналізу сучасних методів у роботі розроблена комплексна методика, яка відрізняється від відомих тим, що враховує і інструменти виявлення, аналізу та оцінювання вразливостей веб-додатків в комплексі з урахуванням штучного інтелекту.

За результатами дослідження запропоновані рекомендації постачальникам щодо оброблення вразливостей веб-додатків та обміну досвідом із застосуванням штучного інтелекту.

Надані конкретні пропозиції та рекомендації щодо практичного впровадження розробленої комплексної методики в реальному середовищі. Розглянуті аспекти інтеграції методики в процес розробки та підтримки веб-додатків, а також визначені оптимальні стратегії впровадження з урахуванням конкретних вимог та характеристик організації

***Галузь застосування*.** Практичне впровадження розробленої методики для організацій і підприємств може значно підвищити рівень безпеки веб-додатків, сприяючи уникненню атак та зменшенню ризиків для користувачів та організацій. Результати даної роботи можуть слугувати підґрунтям для подальших досліджень та вдосконалення стратегій захисту веб-додатків у змінних умовах кіберзагроз.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА БЕЗПЕКА, ІНТЕРНЕТ-ТЕХНОЛОГІЇ, ВЕБ-ДОДАТКИ, МЕТОДИ ТА ЗАСОБИ ОЦІНЮВАННЯ, АНАЛІЗ ВРАЗЛИВОСТЕЙ, СТАНДАРТИ, ШТУЧНИЙ ІНТЕЛЕКТ

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 66 pages, 27 fig., 63 sources.

The purpose of the work is to identify and develop new technologies for effective detection, analysis and evaluation of web application vulnerabilities in order to increase their security level.

The object of the research is the technology of detection, analysis and assessment of web application vulnerabilities.

The subject of the study is the effectiveness of technologies for detecting, analyzing and assessing web application vulnerabilities.

Research methods. Research methods are used to solve the problems: analysis and synthesis in the study of scientific works devoted to the organization of security of web applications; modeling in the development of a complex method of detection, analysis and evaluation of vulnerabilities of modern web applications;

Brief content of the work. The theoretical aspects of the security of web applications were analyzed, their characteristics and structure determined, and the requirements of international standards for protection made it possible to establish an urgent need for the analysis of technologies for detecting and assessing vulnerabilities in connection with the rapid development of modern information technologies.

It was found that modern methods of detecting vulnerabilities do not always satisfy the requirements of effective security of web resources and require the search for new approaches to the organization of information security of web resources.

It has been established that one of the ways to increase the effectiveness of web security is the development of complex methods for detecting, analyzing and evaluating the vulnerabilities of web applications, which should be based on the use of artificial intelligence, neural networks, and machine learning.

Improved evaluation criteria make it possible to establish, in accordance with international standards, the effectiveness of developed web application security

strategies, taking into account all security requirements.

On the basis of the analysis of modern methods, a comprehensive methodology was developed in the work, which differs from the known ones in that it also takes into account the tools for detecting, analyzing and evaluating the vulnerabilities of web applications in a complex, taking into account artificial intelligence.

Based on the results of the study, recommendations are offered to suppliers regarding the handling of web application vulnerabilities and the exchange of experience with the use of artificial intelligence.

Specific proposals and recommendations regarding the practical implementation of the developed complex methodology in a real environment are given. The aspects of the integration of the methodology into the process of development and support of web applications are considered, as well as the optimal implementation strategies are determined, taking into account the specific requirements and characteristics of the organization

Field of application. Practical implementation of the developed methodology for organizations and enterprises can significantly increase the level of security of web applications, helping to avoid attacks and reducing risks for users and organizations. The results of this work can serve as a basis for further research and improvement of web application protection strategies in changing conditions of cyber threats.

KEYWORDS: INFORMATION SECURITY, INTERNET TECHNOLOGIES, WEB APPLICATIONS, ASSESSMENT METHODS AND TOOLS, VULNERABILITY ANALYSIS, STANDARDS, ARTIFICIAL INTELLIGENCE

ЗМІСТ

ВСТУП

Актуальність теми. Величезне значення Інтернету для сучасних підприємств і супутнє зростання складності, частоти та впливу кібератак зробили веб-безпеку критично важливою для безперервності бізнесу. Це перша лінія захисту від загроз, які можуть призвести до розкриття конфіденційних даних, дорогих викупів, репутаційної шкоди, порушень відповідності та безлічі інших наслідків. Інтернет-загрози, які колись були сферою діяльності переважно дрібних хакерів, перетворилися на масштабний бізнес на чорному ринку, який торкається світу організованої злочинності, а також спонсорованого державою шпигунства та саботажу. Деякі з новітніх загроз неймовірно складні, здатні легко обдурити або обійти застарілу безпеку. Крім того, маючи безліч готових інструментів, наборів експлойтів, модулів JavaScript і навіть повністю розроблених програм на продаж, навіть початківець може легко запустити атаку.

Метою безпеки веб-сайтів є запобігання цим (або будь-яким) видам атак. Більш формальне визначення безпеки веб-сайтів – це дія/практика захисту веб-сайтів від несанкціонованого доступу, використання, модифікації, знищення або порушення.

З огляду на зазначене тема кваліфікаційної роботи є актуальною, а використання її результатів сприятиме підвищенню рівня безпеки веб-ресурсів та інформаційної безпеки в цілому.

Методи дослідження. Для вирішення завдань застосовані методи дослідження: аналіз і синтез при дослідженні наукових робіт, присвячених організації безпеки веб-додатків; моделювання при розробленні комплексної методики виявлення, аналізу та оцінюванні вразливостей сучасних веб-додатків;

Об'єктом дослідження є технології виявлення, аналізу та оцінювання вразливостей веб-додатків.

Предмет дослідження – ефективність технологій виявлення, аналізу та оцінювання вразливостей веб-додатків.

Метою роботи є визначення та розробка нових технологій ефективного

виявлення, аналізу та оцінювання вразливостей веб-додатків для підвищення рівня їхньої безпеки.

Для досягнення цієї мети в роботі виконано наступні *завдання:*

1. Здійснений аналіз наукової літератури та публікацій, пов'язаних із виявленням, аналізом та оцінюванням вразливостей веб-додатків.

2. Досліджений сучасний стан вирішення проблеми захисту веб-додатків та зробити аналіз методів виявлення та оцінювання вразливостей веб-додатків.

3. На основі аналізу розроблений алгоритм та комплексна методика, яка включає виявлення, аналіз та оцінювання вразливостей сучасних веб-додатків з використанням штучного інтелекту.

4. Здійснений моніторинг та оцінка ефективності розробленої методики та відпрацьовані пропозиції з практичного застосування.

Методи дослідження. Для вирішення завдань застосовані методи дослідження: аналіз і синтез при дослідженні наукових робіт, присвячених організації безпеки веб-додатків; моделювання при розробленні комплексної методики виявлення, аналізу та оцінюванні вразливостей сучасних веб-додатків.

Наукова новизна одержаних результатів. Розроблена вперше комплексна методика виявлення аналізу та оцінювання вразливості сучасних веб-додатків, яка поєднує вимоги міжнародних та вітчизняних нормативно-правових актів, комплексне застосування сучасних механізмів виявлення, аналізу та оцінювання загроз веб-застосункам.

Удосконалені способи застосування штучного інтелекту для підвищення ефективності безпеки веб-додатків шляхом створення системи обміну досвідом реагування на вразливості веб-додатків користувачів та виконавців.

Практичне значення одержаних результатів. Практичне впровадження розробленої методики може значно підвищити рівень безпеки веб-додатків, сприяючи уникненню атак та зменшенню ризиків для користувачів та організацій. Результати даної роботи можуть слугувати підґрунтям для подальших досліджень та вдосконалення стратегій захисту веб-додатків у змінних умовах кіберзагроз.

Апробація результатів кваліфікаційної роботи: було оприлюднено на Всеукраїнській науковій конференції «Стратегії кіберстійкості: управління ризиками та безперервність бізнесу» 24 лютого 2022 року. Київ. ННІЗІ, ДУТ..

РОЗДІЛ 1
ТЕОРЕТИЧНІ АСПЕКТИ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

1.1 Основні поняття щодо безпеки веб-додатків

У сучасному цифровому світі актуальність веб-безпеки залишається вельми високою. З кожним роком зростає кількість загроз та кібератак, спрямованих на веб-сервери, програми та користувачів. Зловмисники постійно розробляють нові методи атак, використовуючи різноманітні техніки, такі як SQL-ін'єкції, крос-сайт скриптинг, фішинг, DDoS-атаки та інші.

З розвитком Інтернету речей з'являються нові виклики у сфері безпеки, оскільки багато пристроїв підключені до мережі. Несправна конфігурація та недостатній захист можуть призвести до серйозних проблем.

Веб-додаток (або веб-застосунок) - це програмне забезпечення, призначене для роботи на веб-сервері та доступне користувачам через веб-браузер. Веб-додатки взаємодіють з користувачем за допомогою веб-інтерфейсу, який може бути доступний через різні пристрої, такі як комп'ютери, планшети або смартфони [1].

Веб-додатки - це програми, які можна відкрити за допомогою будь-якого браузера. Вони можуть бути розроблені за допомогою різних технологій, таких як HTML, CSS, JavaScript, Python, PHP, Ruby, Java, тощо. Веб-додатки можуть бути класифіковані за різними критеріями, наприклад, за типом взаємодії з користувачем, за типом використовуваної архітектури, за типом використовуваної технології, тощо. Основна мета веб-додатків - надавати користувачам можливість виконувати різноманітні завдання та операції через інтернет [2].

За загальною класифікацією веб-додатки поділяються на [3]:

- односторінкові додатки (SPA) - це додатки, які не перезавантажують сторінку під час взаємодії з користувачем. Вони використовують AJAX для

динамічного оновлення вмісту на сторінці. SPA зазвичай мають більш інтерактивний і швидкий інтерфейс, ніж традиційні веб-додатки;

- багатосторінкові веб-додатки (MPA) - це додатки, які мають більше однієї сторінки. Кожна сторінка містить окремий контент, і користувач може переходити з однієї сторінки на іншу, щоб отримати доступ до різних функцій додатку;

- прогресивні веб-додатки (PWA) - це додатки, які поєднують в собі переваги веб-додатків та мобільних додатків. Вони можуть працювати в автономному режимі, мати доступ до апаратного забезпечення пристрою, використовувати push-сповіщення, тощо.

Основні вагомі види веб-додатків показані на рис. 1.1.

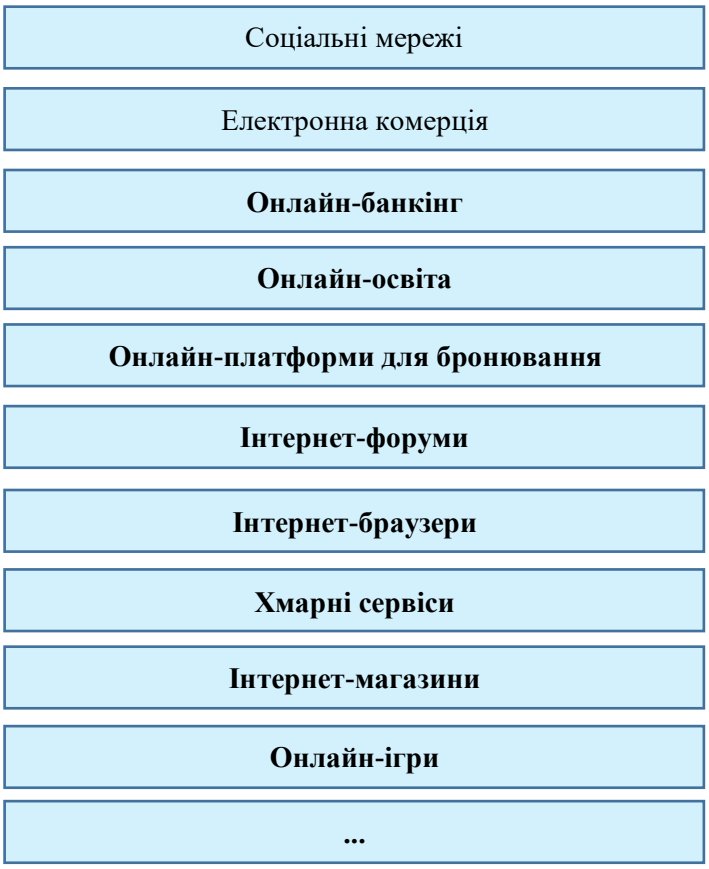

Рис.1.1. Основні види веб-додатків

1. *Соціальні мережі* - це додатки, які дозволяють користувачам обмінюватися повідомленнями, фотографіями, відео, тощо. Деякі з найпопулярніших соціальних мереж включають Facebook, Twitter, Instagram, LinkedIn, тощо.

2. *Електронна комерція* - це додатки, які дозволяють користувачам купувати товари та послуги в Інтернеті. Деякі з найпопулярніших платформ електронної комерції включають Amazon, eBay, Alibaba, тощо.

3. *Онлайн-банкінг* - це додатки, які дозволяють користувачам здійснювати банківські операції в Інтернеті. Деякі з найпопулярніших онлайн-банків включають Chase, Bank of America, Wells Fargo, тощо.

4. *Онлайн-освіта* - це додатки, які дозволяють безкоштовно або на платній основі чи з деякими обмеженнями організувати дистанційне навчання, їх неповний перелік на рис.1.2:

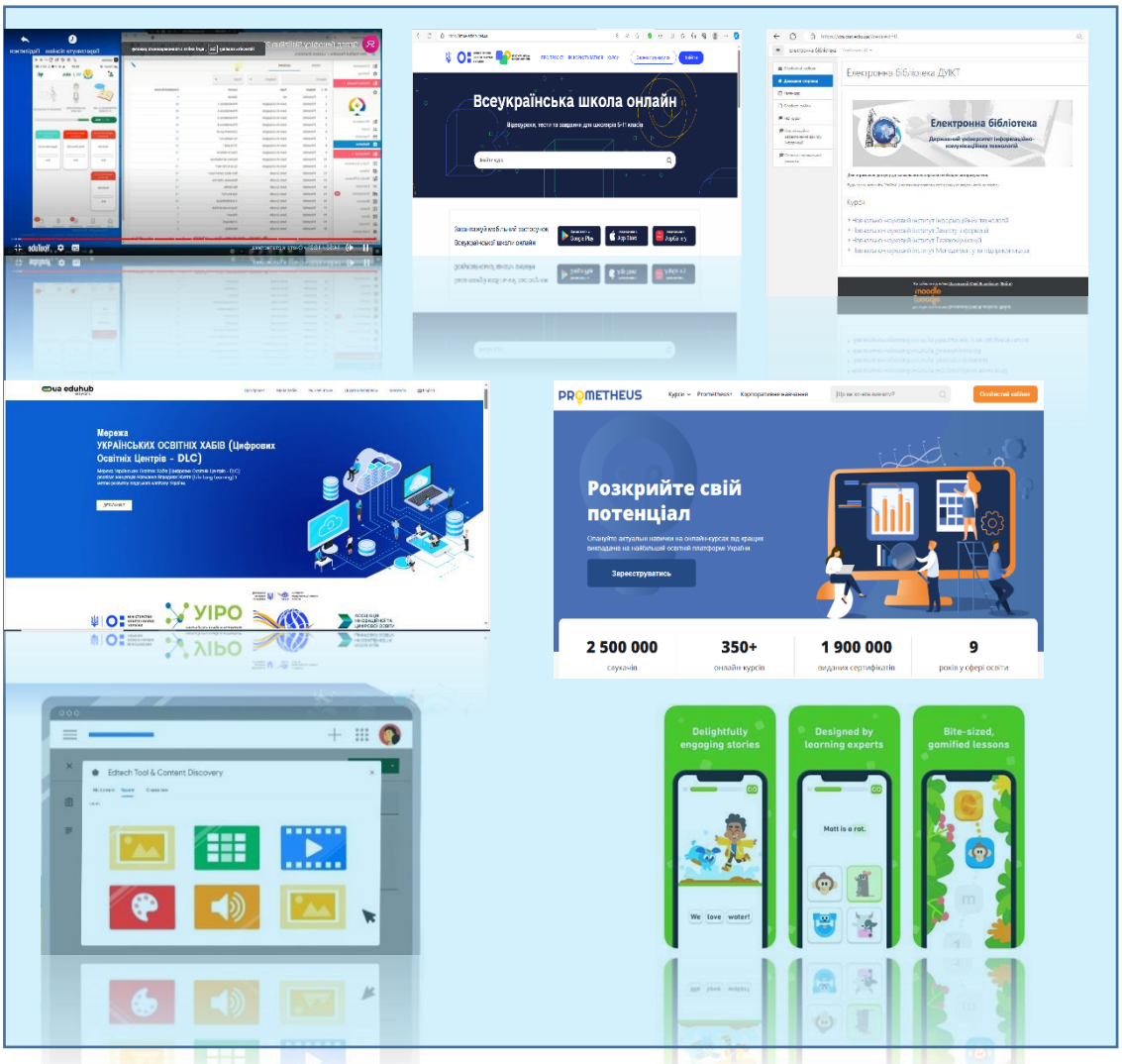

Рис.1.2. Веб-додатки онлайн-освіти

- SkillzRun - мобільний додаток для навчання, який дозволяє створювати власні навчальні програми та курси;
- всеукраїнська школа онлайн - онлайн-платформа, яка надає доступ до навчальних матеріалів для учнів 5-11 класів;
- EduHub - онлайн-платформа для самоосвіти, яка містить безкоштовні та платні курси з різних галузей;
- Moodle - відкрите програмне забезпечення для управління навчанням та системи управління навчанням (LMS). Цей веб-додаток спроектований для надання засобів електронного навчання, де викладачі можуть створювати курси, завдання, форуми та інші ресурси для навчання та оцінювання студентів. Учні можуть взаємодіяти з матеріалами курсів, виконувати завдання, спілкуватися з іншими учнями та отримувати оцінки через цю платформу. Moodle побудований на веб-технологіях, таких як PHP та MySQL, і використовується в основному у навчальних закладах та підприємствах для навчання та управління навчанням. Таким чином, він відповідає визначенню веб-додатка, який дозволяє користувачам отримувати доступ до функціоналу через інтерфейс веб-браузера;
- Prometheus - це платформа, на якій можна отримати актуальні навички на онлайн-курсах отримуючи сертифікати, тут можна обрати курс від Prometheus. Десятки безкоштовних і платних варіантів, багато курсів гарантують працевлаштування для кращих учнів.
- Google Classroom - це веб-платформа, розроблена Google, для організації навчальних процесів. Викладачі можуть створювати курси, розміщувати матеріали, завдання та ресурси для студентів. Студенти та викладачі отримують доступ до Google Classroom через веб-браузер, де вони можуть взаємодіяти з вмістом та спілкуватися;
- Duolingo - це веб-платформа для вивчення мов. Вона надає користувачам доступ до різноманітних вправ, завдань та уроків для вивчення іноземних мов. Користувачі можуть використовувати Duolingo через веб-браузер для отримання уроків та взаємодії з відповідним вмістом…

5. *Онлайн-платформи для бронювання* - це додатки, які дозволяють користувачам забронювати готелі, авіаквитки, автомобілі, тощо. Деякі з

найпопулярніших онлайн-платформ для бронювання включають Booking.com, Expedia, Kayak, тощо.

6. *Відео-стрімінгові сервіси* - це додатки, які дозволяють користувачам дивитися фільми, телешоу, відеокліпи, тощо. Деякі з найпопулярніших відео-стрімінгових сервісів включають Netflix, Hulu, Amazon Prime Video, тощо.

7. *Інтернет-форуми* - це додатки, які дозволяють користувачам обговорювати різні теми, обмінюватися думками та ідеями, тощо. Деякі з найпопулярніших інтернет-форумів включають Reddit, Quora, Stack Overflow, тощо.

8. *Інтернет-браузери* - це додатки, які дозволяють користувачам переглядати веб-сторінки. Деякі з найпопулярніших інтернет-браузерів включають Google Chrome, Mozilla Firefox, Safari, тощо.

9. *Хмарні сервіси* - це додатки, які дозволяють користувачам зберігати, редагувати та обмінюватися даними в Інтернеті. Деякі з найпопулярніших хмарних сервісів включають Google Drive, Dropbox, Microsoft OneDrive, тощо.

10. *Інтернет-магазини* - це додатки, які дозволяють користувачам купувати товари та послуги в Інтернеті. Деякі з найпопулярніших інтернет-магазинів включають Amazon, eBay, Alibaba, тощо.

11. *Онлайн-ігри* - це додатки, які дозволяють користувачам грати в ігри в Інтернеті. Деякі з найпопулярніших онлайн-ігор включають Fortnite, Minecraft, World of Warcraft, тощо.

Запровадження нових технологій, таких як хмарні сервіси, мобільні додатки та інші, вимагає відповідних заходів безпеки, оскільки ці технології можуть використовуватися як вектори атак.

Захист особистої інформації користувачів, включаючи дані про платіжні картки, паролі та інші конфіденційні дані, залишається надто важливою задачею. Веб-сайти та веб-додатки, як зазначає переважна більшість дослідників [5-8] повинні бути належним чином захищені від несанкціонованого доступу. Для компаній і бізнесів важливо захищати свої веб-ресурси від атак, щоб уникнути фінансових втрат, втрати репутації та порушення ділової діяльності.

Безпека веб-додатків - це широкий і складний аспект, який включає в себе різні аспекти. Основні поняття щодо безпеки веб-додатків включають [9-12]:

- автентифікація - це процес підтвердження ідентичності користувача. Важливо використовувати надійні механізми автентифікації, такі як паролі, двоетапна автентифікація (2FA), біометричні дані, поведінкова автентифікація тощо.
- авторизація - визначення рівня доступу до різних веб-ресурсів;
- шифрування даних використовують під час їх передачі між користувачем і веб-сервером (SSL/TLS) і зберігають чутливу інформацію в зашифрованому вигляді на сервері;
- захист від крос-сайт атак (XSS - атаки, при яких зловмисник вбудовує в шкідливий код на веб-сторінку, що виконується в браузері користувача). Заходи безпеки при цьому включають в себе фільтрацію та екранування введених даних;
- захист від крос-сайт скриптінгу (CSRF - атаки, які використовують неправомірно виведені на сторінці користувача запити до веб-додатка). Заходи безпеки включають в себе використання токенів і перевірку походження запитів;
- захист від SQL-ін'єкцій (SQL-ін'єкції - атаки, при яких зловмисник вводить шкідливі SQL-запити в веб-форми або URL-параметри для виклику неправомірних операцій в базі даних). Використання параметризованих запитів та обробка даних від користувачів з врахуванням безпеки є ключовими.
- безпека файлів при завантаженні, зберіганні і обробці файлів користувачами. Важливо перевіряти та обмежувати типи файлів, які можна завантажити, а також використовувати антивірусні сканери;
- моніторинг веб-додатків та журналювання подій для вчасного виявлення та відповіді на можливі загрози безпеки;
- безпека сесій – це використання безпечних механізмів сесій, таких як токени автентифікації та захищені cookies, для уникнення атак на перехоплення сесій;
- обмеження прав доступу: за принципом найменших привілеїв - користувачеві мають видаватися лише ті права, які є абсолютно необхідними для

виконання його функцій.

Ці поняття є лише частиною комплексу заходів забезпечення безпеки веб-додатків, і їхнє правильне впровадження допомагає захистити веб-додатки від різноманітних загроз.

Проведений аналіз [13] виявив величину ризику вразливості, який залежить від того, платформу (рис.1.3) та яку мову програмування використовували для розроблення веб-додатків (рис. 1.4)

Рис. 1.3. Відсотки вразливості веб-додатків

PHP	Частка сайтів, %	Java	Частка сайтів, %	ASP.NET	Частка сайтів, %
Cross Site Scripting	90	Cross Site Scripting	80	Cross Site Scripting	73
Credential/Session Prediction	86	Fingerprinting	60	Brute Force	73
Brute Force	81	Brute Force	45	Fingerprinting	55
Information Leakage	67	Credential/Session Prediction	45	Cross Site Request Forgery	55
SQL Injection	62	Server Misconfiguration	35	Credential/Session Prediction	45
Fingerprinting	43	Information Leakage	30	Information Leakage	45

Рис.1.4. Показники схильності сайтів до атак

Веб-додатки є веб-ресурсом і самі можуть використовувати веб-ресурси, які теж піддаються впливам загроз і є потенційною небезпекою, тому необхідно визначитись і з цими поняттями.

Веб-ресурс - це певний вузол чи точка, куди включений спеціальний ідентифікатор, що дозволяє за необхідності легко знайти потрібну сторінку на просторах мережі Інтернет. Найчастіше один домен відповідає одному веб-ресурсу, проте іноді буває так, що на одному домені розміщуються кілька веб-ресурсів, або один веб-ресурс має для себе кілька доменів.

Існує кілька видів веб-ресурсів: відкриті; напіввідкриті; закриті.

На відкритих ресурсах користувачам доступні всі сервіси, чого не можна сказати про напіввідкриті, де потрібна реєстрація, та закриті, куди можна потрапити тільки за запрошенням. Крім того, виділяють іще локальні ресурси, які розміщені в зоні локальної мережі.

За способом представлення інформації розділяють на [14-15]:

- інформаційні ресурси – тематичні сайти і портали, ще в наявності інформація вузької направленості;
- інтернет-представництва – сайти бізнесу (реклама, візитки, інтернет-магазини тощо);
- веб-сервіси для виконання різного роду завдань в Інтернеті (блоги, пошукові, хостінги …);

Веб-ресурс - це будь-який з ресурсів, які створюються під час розробки веб-додатку, наприклад, веб-проекти, HTML-сторінки, JSP-файли, сервлети, користувальницькі бібліотеки тегів і архівні файли (рис.1.5). Безпека веб-ресурсів є найважливішим аспектом онлайн-безпеки. Три фундаментальні концепції безпеки, важливі для інформації в Інтернеті, – це конфіденційність, цілісність і доступність. Конфіденційність означає контроль, хто може читати інформацію, цілісність гарантує, що інформація та програми змінюються лише визначеним і дозволеним способом, а доступність гарантує що авторизовані користувачі мають безперервний доступ до інформації та ресурсів.

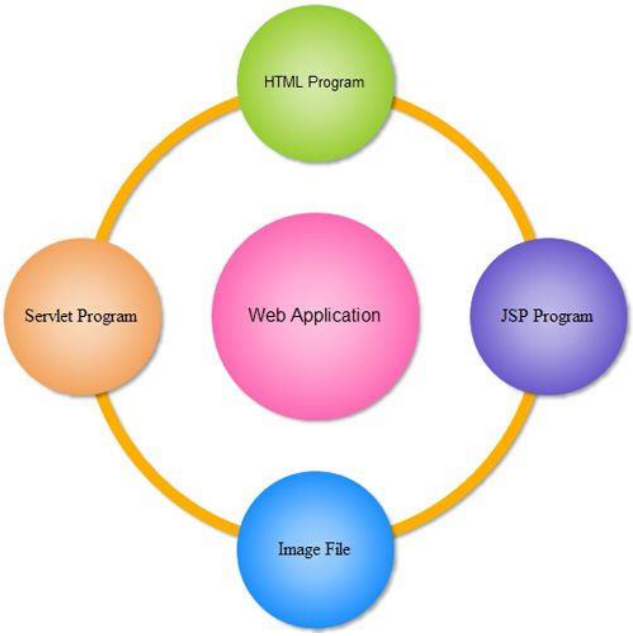

Рис. 1.5. Веб-ресурси, які створюються під час розробки веб-додатків

Веб-безпека – це широка категорія рішень безпеки, які захищають користувачів, пристрої та мережу від кібератак в Інтернеті, таких як зловмисне програмне забезпечення, фішинг тощо, які можуть призвести до зламу та втрати даних. Це не лише захист веб-сайту , це також про захист усієї вашої мережі. Безпечний веб-сайт захищає інформацію та забезпечує безпеку користувачів.

Веб-безпека означає використання новітніх протоколів безпеки, надання послуг через безпечне з'єднання та впровадження засобів контролю, які гарантують, що потрібна особа з потрібними привілеями може отримати доступ до потрібної інформації в потрібний час (відоме як керування ідентифікацією, обліковими даними та доступом.

Веб-безпека є складовою кібербезпеки, основні поняття і визначення якої зазначені в нормативно-правових документах [16-19]. Кібербезпека включає широкий спектр заходів та практик, спрямованих на захист інформації, систем, мереж та інших цифрових активів від різноманітних загроз і атак [20].

Веб-безпека, з іншого боку, фокусується на захисті веб-додатків, веб-сайтів та пов'язаних сервісів від загроз, що можуть виникнути внаслідок вразливостей у веб-технологіях, програмному забезпеченні та протоколах. Оскільки веб-додатки стають все більш важливою частиною діяльності бізнесу

та комунікації, веб-безпека стає важливою галуззю кібербезпеки.

За оцінками Cybersecurity Ventures, до 2025 року глобальна кіберзлочинність коштуватиме 10,5 трильйонів доларів США на рік – це більший прибуток, ніж найбільша у світі незаконна торгівля наркотиками – і половина світових даних буде зберігатися в хмарі. З огляду на те, що поставлено на карту, легко зрозуміти, чому ефективна веб-безпека така важлива сьогодні [21].

Ефективна безпека любого веб-сайту при проектуванні веб-додатку вимагає зусиль захисту всього веб-сайту та оточення: у самому веб-додатку, конфігурації веб-сервера, власних політик створення та поновлення паролів, а також коду на стороні клієнта. Хоча все це звучить дуже зловісно, хороша новина полягає в тому, що якщо використовувати серверний веб-фреймворк, він майже напевно забезпечить «за замовчуванням» надійні та добре продумані механізми захисту від ряду найбільш поширених атак. Інші атаки можна пом'якшити за допомогою конфігурації веб-сервера, наприклад, увімкнувши HTTPS. Нарешті, існують загальнодоступні інструменти сканування вразливостей, які можуть допомогти з'ясувати, чи не припустилися які-небуть очевидні помилки [22].

Сьогодні перед дослідниками стоїть актуальне завдання розгляду ключових аспектів, які впливають на успішність управлінської діяльності в галузі кібербезпеки, а також визначення оптимальних стратегій та практик, спрямованих на забезпечення високого рівня захисту інформаційних ресурсів.

Веб-безпека відкриває широку мережу для захисту користувачів і кінцевих точок від шкідливих електронних листів, зашифрованих загроз, шкідливих або скомпрометованих веб-сайтів і баз даних, зловмисних перенаправлень, викрадення тощо [23-24]. Найпоширеніші загрози для веб-безпеки визначені на рис. 1.6.

Програми-вимагачі: ці атаки шифрують дані, а потім вимагають викуп в обмін на ключ розшифровки. Під час атаки подвійного вимагання ваші дані також викрадаються.

Існує незліченна кількість варіантів *шкідливого програмного забезпечення*, яке може призвести до чого завгодно: від витоку даних, шпигунства та

несанкціонованого доступу до блокувань, помилок і збоїв системи.

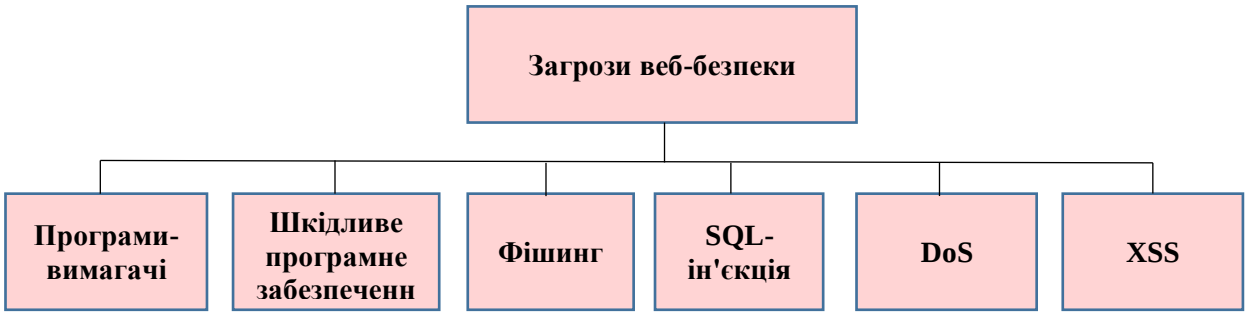

Рис.1.6. Основний перелік найпоширеніших загроз

Фішинг: ці атаки, які часто здійснюються електронною поштою, текстовими повідомленнями або шкідливими веб-сайтами, обманом змушують користувачів розголошувати облікові дані для входу або завантажувати шпигунське програмне забезпечення.

SQL-ін'єкція: ці атаки використовують уразливість вхідного сигналу на сервері бази даних, дозволяючи зловмиснику виконувати команди, які дозволяють йому отримувати, маніпулювати або видаляти дані.

Відмова в обслуговуванні (DoS): ці атаки уповільнюють або навіть вимикають мережевий пристрій, наприклад сервер, надсилаючи йому більше даних, ніж він може обробити. У розподіленій DoS-атаці, тобто DDoS-атаці, це здійснюється багатьма викраденими пристроями одночасно.

Міжсайтовий скриптинг (XSS): у цьому типі ін'єкційної атаки зловмисник вводить шкідливий код на надійний веб-сайт, вводячи його в незахищене поле введення користувача.

Ідеальне рішення для веб-безпеки використовує кілька технологій для зупинки зловмисного програмного забезпечення та програм-вимагачів, блокування фішингових доменів, обмеження використання облікових даних тощо, створюючи цілісний захист.

1.2 Характеристики та структура WEB-додатків

Основні характеристики веб-додатків включають (рис.1.7):

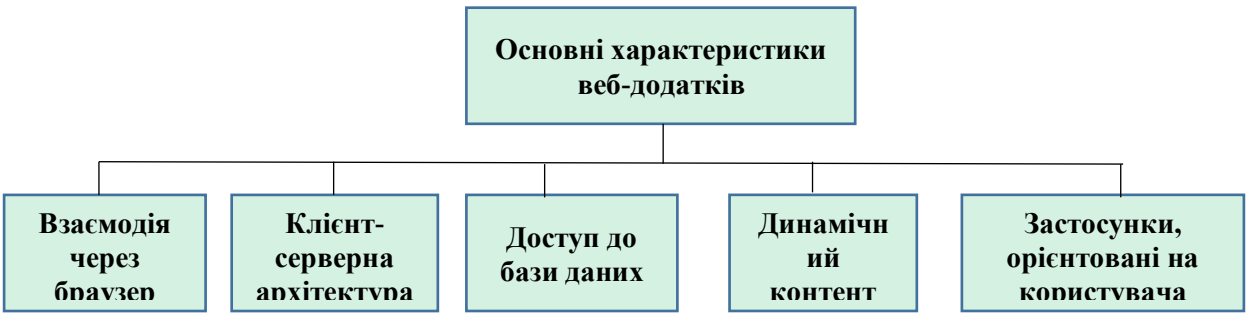

Рис. 1.7. Основні характеристики веб-додатків

Взаємодія через браузер: користувачі отримують доступ до веб-додатків через веб-браузер, такий як Google Chrome, Mozilla Firefox, або Safari. Вони не вимагають встановлення додаткового програмного забезпечення на пристрої користувачів.

Клієнт-серверна архітектура: веб-додатки працюють на основі клієнт-серверної моделі, де клієнтська частина (веб-браузер) отримує та відображає дані, а серверна частина обробляє запити та надає відповіді.

Доступ до бази даних: багато веб-додатків використовують бази даних для зберігання та управління інформацією. Це може бути даних користувачів, контенту, конфігурацій тощо.

Динамічний контент: веб-додатки можуть генерувати та відображати динамічний контент в залежності від введення користувача, контексту та інших параметрів.

Застосунки, орієнтовані на користувача зазвичай забезпечують інтерфейс, який дозволяє користувачам взаємодіяти з додатком, виконувати завдання та налаштовувати параметри відповідно до своїх потреб.

Багато характеристик веб-додатків, таких як надійність, безпека, масштабованість, швидкість відгуку, залежить від архітектури веб-додатків, з якою ви вирішили працювати. Правильна архітектура веб-додатків прокладає шлях для майбутніх планів розширення та масштабованості. Тому завжди

корисно вивчити вимоги та цілі, перш ніж хтось почне процес розробки програми.

Архітектура веб-додатків - це механізм, який дає нам роз'яснення того, як встановлюється з'єднання між клієнтом і сервером. Він визначає, як компоненти програми взаємодіють один з одним [24]. Неважливо, який розмір і рівень складності програми, всі вони працюють за одним принципом, тільки деталі можуть відрізнятися.

З технічної точки зору, коли користувач робить запит на веб-сайті, різні компоненти додатків, інтерфейси користувача, системи проміжного програмного забезпечення, бази даних, сервери та браузер взаємодіють один з одним. Архітектура веб-додатків - це структура, яка пов'язує ці відносини воєдино і підтримує взаємодію між цими компонентами.

Коли користувач взаємодіє з веб-сайтом і отримує відповідь з боку сервера, весь процес виконується протягом декількох секунд. Найголовніше, на що ми повинні звернути увагу, це код, який був переданий браузеру. Цей код може мати або не мати конкретних інструкцій, які вказують браузеру, як реагувати на різні типи введених користувачем даних. Ось чому архітектура веб-додатків включає в себе всі підкомпоненти і взаємозаміни зовнішніх додатків для цілого програмного додатку. Архітектура веб-додатків повинна мати справу з надійністю, масштабованістю, безпекою та стійкістю через велику кількість глобального мережевого трафіку.

Всі веб-додатки працюють на стороні клієнта і сервера. Коли користувач робить запит, з обох сторін запускаються в основному дві програми:

- код, який запускається в браузері та працює відповідно до введених користувачем даних;
- код на сервері, який відповідає на HTTP-запити

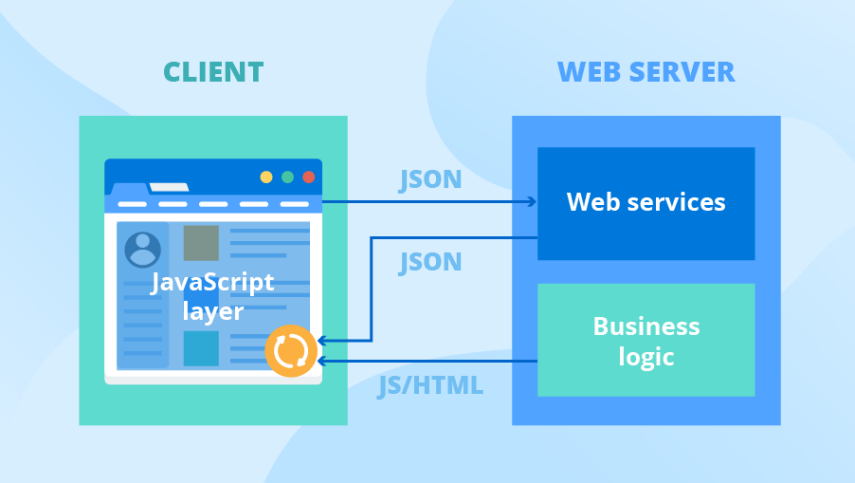

Рис. 1.8. Клієнт-серверна взаємодія

Серверний код може бути написаний за допомогою мов Python, JavaScript, C#, PHP, Ruby on Rails тощо. Будь-який код може мати можливість працювати на сервері, якщо він може відповідати на HTTP-запити. Код на стороні сервера в основному відповідає за створення сторінки, яку запитав користувач. Він також зберігає різні типи даних, такі як профілі користувачів, твіти, сторінки тощо. Код на стороні сервера не може бути видимий кінцевим користувачем (за винятком рідкісної несправності)

Мови на стороні клієнта включають комбінацію HTML, CSS і JavaScript. Цей код аналізується браузером, і користувач може його бачити, а також редагувати. Тільки за допомогою HTTP-запитів клієнтський код може обмінюватися даними з сервером. Крім того, він не може зчитувати файли безпосередньо з сервера.

Компоненти архітектури веб-додатків розкриті в роботі [25-26].

Архітектура веб-додатків працює на різних компонентах. Ці компоненти можна розділити на два напрямки.

1. Компоненти програми інтерфейсу користувача. Як випливає з назви, ця категорія набагато більше пов'язана з інтерфейсом/досвідом користувача. У цій категорії роль веб-сторінки пов'язана з відображенням, інформаційними панелями, журналами, повідомленнями, статистикою, налаштуваннями конфігурації тощо, і вона не має нічого спільного з функціональністю чи роботою веб-програми.

2. Структурні компоненти: ця категорія в основному пов'язана з функціональністю веб-додатку, з яким взаємодіє користувач, контролем і сховищем бази даних. Як випливає з назви, мова йде набагато більше про структурну частину веб-додатку. Ця структурна частина включає в себе:

- веб-браузер або клієнт;
- сервер веб-додатків;
- сервер баз даних.

Архітектурні патерни веб-додатків розділені на безліч різних шарів або рівнів, що називається багато- або трирівневою архітектурою. Ви можете легко замінити та оновити кожен шар самостійно.

Трирівнева архітектура веб-додатків [26] (рис.1.9).

Рис.1.9. Трирівнева архітектура «клієнт-сервер»

Презентаційний шар: цей рівень доступний клієнту через браузер і включає компоненти інтерфейсу користувача та компоненти процесу інтерфейсу користувача. Ці компоненти інтерфейсу користувача побудовані за допомогою HTML, CSS і JavaScript (і їх фреймворків або бібліотеки), де кожен з них відіграє різну роль у створенні інтерфейсу користувача.

Бізнес-рівень - його також називають бізнес-логікою або логікою домену або прикладним рівнем. Він приймає запит користувача від браузера, обробляє його та регулює маршрути, через які буде здійснюватися доступ до даних. Весь робочий процес закодований у цьому шарі. Можна взяти приклад бронювання готелю на сайті. Мандрівник пройде через послідовність подій, щоб забронювати

номер у готелі, а весь робочий процес візьме на себе бізнес-логіка.

Шар стійкості - його також називають рівнем зберігання або доступу до даних. Цей рівень збирає всі виклики даних і надає доступ до постійного сховища програми. Бізнес-рівень тісно пов'язаний з рівнем стійкості, тому логіка знає, з якою базою даних спілкуватися, і процес отримання даних стає більш оптимізованим. Сервер і програмне забезпечення системи управління базами даних існують в інфраструктурі зберігання даних, яка використовується для зв'язку з самою базою даних, додатками та інтерфейсами користувача для отримання даних і їх аналізу. Ви можете зберігати дані на апаратних серверах або в хмарі.

Є деякі інші частини веб-додатку, які відокремлені від основних шарів, що існують в архітектурі:

наскрізний код - ця частина відповідає за комунікації, оперативне управління та безпеку. Він впливає на всі частини системи, але ніколи не повинен змішуватися з ними.

сторонні інтеграції - використовуючи сторонні API (англ. Application Programming Interface), ми можемо інтегрувати платіжні шлюзи, соціальні логіни, GDS на туристичних сайтах тощо.

Типи архітектури веб-додатків

1. Односторінкові програми: Сьогодні багато сучасних веб-додатків розроблені як односторінкові веб-додатки, які включають лише найнеобхідніші елементи та інформацію для створення інтуїтивно зрозумілого та інтерактивного користувацького досвіду. В односторінковому додатку контент або інформація оновлюється на поточній сторінці, а не завантажується нова сторінка з сервера за кожну дію, виконану користувачем. Програма запитує лише необхідну інформацію про вміст, і це запобігає переривам у роботі користувача.

AJAX, асинхронний JavaScript і XML в основному використовуються для зв'язку між сторінками. Користувачі можуть продовжувати взаємодію зі сторінкою, поки на сторінці оновлюється вміст (швидша взаємодія).

2. Мікросервіси (рис.1.10) це невеликі та легкі сервіси, які виконують

певну, єдину функціональність.

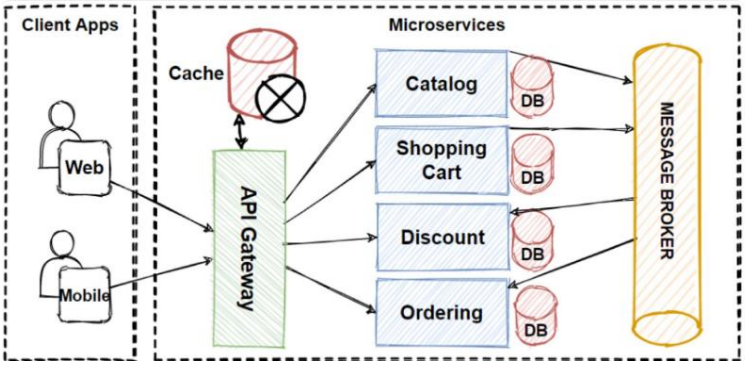

Рис.1.10. Мікросервіси

Компоненти в додатках не залежать один від одного, тому немає необхідності розробляти кожен компонент за допомогою однієї і тієї ж мови програмування. Це дає розробникам гнучкість у виборі мови або стеку технологій на власний вибір. Це підвищує продуктивність розробників і прискорює процес розробки.

3. Безсерверні архітектури (Serverless) [27-28] - цьому підході розробники передають управління сервером та інфраструктурою на аутсорсинг сторонньому постачальнику послуг хмарної інфраструктури (рис. 1.11).

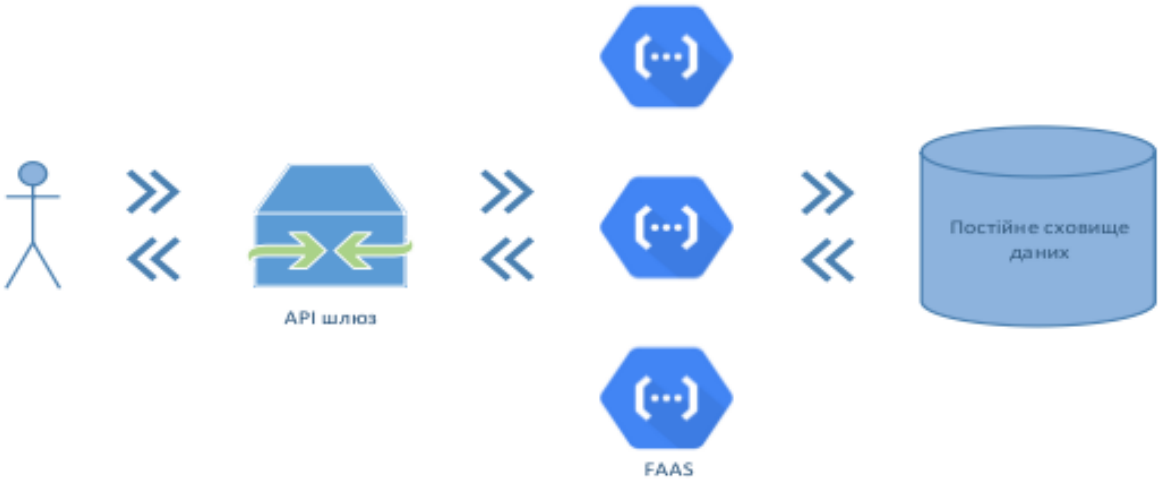

Рис.1.11. Схематичне зображення Serverless архітектури

Перевага цього підходу полягає в тому, що він дозволяє додатку виконувати необхідну або кастомну логіку, не турбуючись про завдання, пов'язані з інфраструктурою. Цьому підходу в основному віддають перевагу компанії, які не хочуть керувати або підтримувати сервери та обладнання, для

якого вони розробили веб-додаток.

1.3 Вимоги міжнародних стандартів щодо захисту WEB-ресурсів

Існує кілька міжнародних стандартів та рекомендацій, які стосуються захисту веб-додатків. Деякі з них виходять від міжнародних організацій, таких як Міжнародна організація зі стандартизації (ISO) та Організація забезпечення безпеки в інтернеті (OWASP - Open Web Application Security Project) [29-30]. Важливо відзначити, що стандарти можуть оновлюватися, тому завжди слід перевіряти найновіші версії.

Деякі важливі стандарти та рекомендації включають:

ISO/IEC 27001: Це стандарт для управління інформаційною безпекою. Він надає загальний фреймворк для впровадження ефективної системи управління інформаційною безпекою, включаючи аспекти веб-безпеки [32].

ISO/IEC 27002 - стандарт надає практичні рекомендації та контрольні заходи щодо імплементації заходів інформаційної безпеки, включаючи аспекти веб-додатків;

OWASP (Open Web Application Security Project) - we не стандарт, але проект, який надає рекомендації та ресурси щодо безпеки веб-додатків. OWASP випускає "Top Ten Project", який перераховує найбільш критичні загрози безпеці веб-додатків та надає поради щодо їх захисту [30,31,33];

NIST SP 800-53 - wей стандарт визначає контрольні заходи для федеральних інформаційних систем та організацій, які працюють з федеральними інформаційними системами в Сполучених Штатах [34];

PCI DSS (Payment Card Industry Data Security Standard) - стандарт безпеки даних індустрії платіжних карт, призначений для організацій, які приймають, обробляють або передають платіжні карти [35].

Дотримання цих стандартів разом із вітчизняними нормативно-правовими документами [36-38] може допомогти організаціям забезпечити адекватний

рівень захисту веб-додатків та знизити ризики кіберзлочинності. Важливо враховувати, що стандарти можуть різнитися в залежності від конкретної галузі та типу даних, які обробляються.

ISO/IEC 27001 та ISO/IEC 27002 – це серія міжнародних стандартів, що визначають вимоги і рекомендації для систем управління інформаційною безпекою (ISMS). Основні вимоги ISO/IEC 27001 і кількох конкретних пунктів з ISO/IEC 27002 наступні:

1) для ISO/IEC 27001:2013 (Системи управління інформаційною безпекою)

Область застосування: організації повинні визначити область застосування ISMS та встановити обґрунтований обсяг.

Політика інформаційної безпеки: розроблення політики інформаційної безпеки, що відображає зобов'язання власника організації.

Ризики та оцінка вразливостей: визначення ризиків і виявлення вразливостей у відносинах із системами інформаційної безпеки.

Менеджмент ризиків: розроблення та впровадження планів ризик-менеджменту для зниження інформаційних ризиків.

Безпека в робочому середовищі: забезпечення безпеки фізичного середовища, включаючи заходи з контролю доступу та виключення зайвих осіб.

2) для ISO/IEC 27002:2013 (Практичні рекомендації щодо управління інформаційною безпекою)

Управління активами: 8.1.1: Інвентаризація активів. 8.1.2: Класифікація інформації.

Управління доступом: 9.1.1: Визначення відповідальності за ідентифікацію та автентифікацію. 9.1.2: Встановлення прав доступу та відкликання прав.

Шифрування: 10.1.1: Вибір алгоритмів і ключів для шифрування.

Фізична та довірлива безпека: 11.1.1: Політика безпеки робочого місця.

Управління інцидентами: 16.1.1: Інструкції з управління інцидентами.

Управління заходами з безпеки: 18.1.1: Створення політики забезпечення безпеки інформації.

Управління документацією та записами: 7.5.1: Визначення і управління документами.

Ці стандарти адресують загальні аспекти управління інформаційною безпекою, але мають вимоги і рекомендації, які застосовуються до веб-додатків. Наприклад, вони включають положення про управління активами, контроль доступу, шифрування та інші аспекти, які можуть бути застосовані у контексті веб-додатків.

Це всього лише декілька конкретних пунктів з обох стандартів. Детальні вимоги і рекомендації можна знайти безпосередньо в тексті стандартів ISO/IEC 27001 і ISO/IEC 27002.

OWASP (ASVS) визначає набір вимог, спрямованих на вдосконалення безпеки веб-додатків. ASVS розділяє вимоги на три рівні, кожен з яких орієнтований на різні потреби безпеки. Загальний огляд кожного рівня та конкретних вимог для кожного рівня наступний:

Рівень 1: Основні вимоги безпеки.

HTTP Security: 1.1: Використання безпечного протоколу (HTTPS). 1.2: Відсутність конфіденційної інформації в URL.

Session Management: 2.1: Захист від атак на сесії. 2.2: Завершення сесії після виходу або бездіяльності.

Data Input Validation and Representation: 3.1: Валідація введених даних. 3.2: Захист від введення скриптів (XSS).

Security Architecture: 4.1: Використання безпечної архітектури. 4.2: Використання безпечних технік мікросервісної архітектури.

Рівень 2: Стандартні вимоги безпеки

Cryptography: 1.1: Використання безпечних криптографічних алгоритмів. 1.2: Захист від атак на сторонні автентифікаційні сервери.

Authentication: 2.1: Використання сильної автентифікації. 2.2: Захист від атак на автентифікацію.

Session Management: 3.1: Валідація та обмеження сесійних токенів. 3.2: Захист від атак на перехоплення сесій.

Access Control: 4.1: Ефективне управління доступом.

Рівень 3: Вимоги для додатків високого рівня

Cryptography: 1.1: Використання ключів і сертифікатів відповідно до найкращих практик.

Authentication: 2.1: Використання сильних механізмів автентифікації. 2.2: Використання безпечних механізмів автентифікації мережі.

Session Management: 3.1: Захист від атак на зміну сесій. 3.2: Захист від атак на створення сесій.

Access Control: 4.1: Контроль доступу на основі ролей та функцій.

Це лише загальний огляд, і сам стандарт має докладний документ, який включає всі вимоги кожного рівня, їх пояснення та приклади. Розробники та аудитори можуть використовувати ASVS для підвищення безпеки веб-додатків на різних рівнях складності та ризику.

NIST SP 800-53 надає загальні вимоги для інформаційної безпеки, включаючи аспекти, які можуть впливати на веб-додатки. Вимоги, такі як управління обліковими записами, контроль доступу, управління інцидентами та інші, застосовуються до веб-середовищ:

AC-2: Account Management.

AC-2.1.1: Account Management Policy and Procedures: Організація повинна розробити, затвердити та впровадити політику та процедури управління обліковими записами.

AC-2.2.1: Automated System Account Management: Системи повинні використовувати автоматизовані засоби для управління обліковими записами, включаючи створення, зміну, блокування та видалення.

AC-3: Access Enforcement.

AC-3.1.1: Session Authenticity: Організація повинна визначити та впровадити механізми для забезпечення автентичності сесій.

AC-3.1.2: Session Timeout: Організація повинна визначити та впровадити механізми для встановлення максимального часу сесії та автоматичного виходу при бездіяльності.

AC-4: Information Flow Enforcement.

AC-4.1.1: Use of Cryptography: Організація повинна визначити та впровадити політику щодо використання криптографії для забезпечення конфіденційності та цілісності даних.

AC-4.2.1: External Information Sharing: Організація повинна визначити та впровадити механізми для контролю потоку інформації при спільному використанні даних зовнішніми сутностями.

AC-6: Least Privilege.

AC-6.1.1: Least Privilege Policy and Procedures: Організація повинна розробити, затвердити та впровадити політику та процедури щодо присвоєння найменшого рівня привілеїв.

AC-6.2.1: Privilege Management: Організація повинна визначити та впровадити механізми для управління привілеями, включаючи надання та відкликання.

AC-7: Unsuccessful Login Attempts.

AC-7.1.1: Unsuccessful Login Attempts: Організація повинна визначити та впровадити механізми для моніторингу та обробки невдалих спроб входу.

ВИСНОВОК до розділу 1. У розділі детально проаналізовано основні терміни та поняття, пов'язані з безпекою веб-додатків. Визначено ключові поняття, такі як автентифікація, авторизація, шифрування, атаки на веб-додатки та інші. Проаналізовані сучасні виклики та тенденції в галузі веб-безпеки. розглянуті характеристики та архітектурна структура веб-додатків. Описано різні компоненти веб-додатків, такі як клієнтська та серверна сторони, бази даних, протоколи комунікації. Проаналізовано важливі аспекти, пов'язані з безпекою на кожному рівні архітектури.

Досліджені вимоги та рекомендації міжнародних стандартів з безпеки веб-додатків. Висвітлено роль таких стандартів, як OWASP, PCI DSS, ISO/IEC 27001, в забезпеченні ефективного захисту веб-додатків. Аналіз внесених стандартами

інновацій та найкращих практик дозволяє забезпечити відповідність веб-додатків вимогам сучасного стандарту безпеки.

визначає теоретичний фундамент для дослідження в галузі безпеки веб-додатків у магістерській роботі. На його основі можна визначити основні виклики, які будуть досліджені у подальших розділах роботи, а також визначити методи та інструменти, які будуть використовуватися для розв'язання конкретних завдань.

За результатами розгляду теоретичних аспектів безпеки веб-додатків встановлено, що:

- розвиток технологій впливає як на організаційні питання захисту веб-ресурсів так і на посилення та вразливостей усіх структурних елементів веб-ресурсів;
- сучасний стан безпеки веб-додатків потребує детального аналізу технологій виявлення та оцінювання вразливостей веб-додатків, пошуку нових шляхів та сучасних методів удосконалення захисту, своєчасного прийняття управлінських рішень у кібербезпеці;
- безпеку веб-додатків необхідно організовувати у відповідності із напрацьованими міжнародними стандартами щодо захисту, враховуючи при цьому національні особливості.

Зроблений аналіз теоретичних аспектів безпеки веб-додатків визначає теоретичний фундамент для дослідження в галузі безпеки веб-додатків у магістерській роботі. На його основі можна визначити основні виклики, які будуть досліджені у подальших розділах роботи, а також визначити методи та інструменти, які будуть використовуватися для розв'язання конкретних завдань.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ, АНАЛІЗУ ТА ОЦІНЮВАННЯ ВРАЗЛИВОСТЕЙ СУЧАСНИХ ВЕБ-ДОДАТКІВ

2.1 Типові вразливості веб-додатків

Веб-додатки можуть бути вразливими до різних атак, і ідентифікація та усунення цих вразливостей є ключовим завданням з точки зору безпеки. Деякі типові вразливості веб-додатків включають [39-42]:

SQL Injection (SQLi) - атаки, під час яких зловмисник вставляє SQL-код у введені дані, що призводить до неправомірного доступу до бази даних;

Cross-Site Scripting (XSS) - зловмисник вбудовує веб-сторінку шкідливий скрипт, який виконується в браузері іншого користувача, що може призвести до викрадення сесій та інших проблем;

Cross-Site Request Forgery (CSRF) - атаки, під час яких зловмисник використовує автентифіковану сесію користувача для виконання неправомірних дій в іншому контексті, наприклад, для відправки запитів в інші веб-додатки;

Security Misconfigurations (Неправильна конфігурація безпеки) - недостатні або неправильно встановлені параметри безпеки, такі як дозвіл неправильних прав доступу, можуть призвести до витоку конфіденційної інформації або інших проблем;

Sensitive Data Exposure (Витік конфіденційної інформації) - неправильне зберігання або передача конфіденційної інформації може стати об'єктом атак і викрадення даних;

Insecure Direct Object References (IDOR) - зловмисники використовують недостатні або відсутні перевірки доступу для отримання доступу до об'єктів чи даних, до яких вони не мають права доступу.

Unvalidated Redirects and Forwards (Непідтверджені перенаправлення та

відправлення) - атаки, під час яких зловмисник може перенаправити користувача на інші сайти або використовувати внутрішні перенаправлення;

Security Headers Missing (Відсутність заголовків безпеки) - відсутність або неправильна конфігурація заголовків безпеки HTTP може збільшити ризик різних атак, таких як XSS$

File Upload Vulnerabilities (Вразливості завантаження файлів): Неправильна обробка або відсутність перевірок при завантаженні файлів може призвести до виконання коду або завантаження шкідливих файлів.

Це лише кілька прикладів вразливостей, і важливо вживати широкий спектр заходів для захисту веб-додатків від різних загроз. Організації також можуть використовувати інструменти сканування безпеки та проводити аудит безпеки для виявлення та усунення цих вразливостей.

Більш детально про міжсайтовий скриптинг (XSS) розкрито в роботах дослідників [43-46].

XSS – це термін, який використовується для опису класу атак, які дозволяють зловмиснику впроваджувати клієнтські скрипти через веб-сайт у браузери інших користувачів. Оскільки введений код надходить у браузер із сайту, код є надійним і може виконувати такі дії, як надсилання файлу cookie авторизації сайту користувача зловмиснику. Коли зловмисник має файл cookie, він може увійти на сайт так, ніби він є користувачем, і зробити все, що може користувач, наприклад, отримати доступ до даних своєї кредитної картки, переглянути контактні дані або змінити паролі. XSS-вразливості історично зустрічаються частіше, ніж будь-які інші типи загроз безпеці. Уразливості XSS поділяються на відображені та постійні, залежно від того, як сайт повертає введені скрипти браузеру.

Відображена XSS-уразливість виникає, коли контент користувача, який передається на сервер, повертається негайно і не змінюється для відображення в браузері. Будь-які скрипти в оригінальному користувацькому контенті будуть запущені під час завантаження нової сторінки. Наприклад, розглянемо функцію пошуку на сайті, де пошукові терміни закодовані як параметри URL-адреси, і ці

терміни відображаються разом із результатами. Зловмисник може створити пошукове посилання, яке містить шкідливий скрипт як параметр (наприклад,), і надіслати його електронною поштою іншому користувачеві. Якщо цільовий користувач перейде за цим «цікавим посиланням», скрипт буде виконаний при відображенні результатів пошуку. Як обговорювалося раніше, це дає зловмиснику всю інформацію, необхідну для входу на сайт як цільового користувача, потенційно здійснюючи покупки як користувач або ділячись своєю контактною інформацією за допомогою скрипта:

http://developer.mozilla.org?q=beer<script%20src="http://example.com/tricky.js"></script>

Постійна XSS-вразливість виникає, коли шкідливий скрипт зберігається на веб-сайті, а потім знову відображається без змін, щоб інші користувачі могли виконати його мимоволі. Наприклад, дошка обговорень, яка приймає коментарі, що містять незмінений HTML, може зберігати шкідливий сценарій зловмисника. Коли коментарі відображаються, скрипт виконується і може відправити зловмиснику інформацію, необхідну для доступу до облікового запису користувача. Цей вид атаки є надзвичайно популярним і потужним, оскільки зловмисник може навіть не мати прямої взаємодії з жертвами.

Хоча дані або запити є найпоширенішим джерелом вразливостей XSS, будь-які дані з браузера є потенційно вразливими, наприклад, дані файлів cookie, що відображаються браузером, або файли користувача, які завантажуються та відображаються методами *POST GET*.

Найкращим захистом від вразливостей XSS є видалення або відключення будь-якої розмітки, яка потенційно може містити інструкції для запуску коду. Для HTML це включає такі елементи, як

<script><object><embed><link>

Процес модифікації даних користувача таким чином, щоб вони не могли бути використані для запуску сценаріїв або іншим чином вплинути на виконання коду сервера, відомий як дезінфекція вхідних даних. Багато веб-фреймворків за

замовчуванням автоматично видаляють введені користувачем дані з HTML-форм.

SQL-ін'єкції. Уразливості SQL-ін'єкцій дозволяють зловмисникам виконувати довільний SQL-код у базі даних, дозволяючи отримувати доступ, змінювати або видаляти дані незалежно від дозволів користувача [47]. Успішна ін'єкційна атака може підробити ідентичності, створити нові профілі з правами адміністрування, отримати доступ до всіх даних на сервері або знищити/змінити дані, щоб зробити їх непридатними для використання. Типи SQL-ін'єкцій включають SQL-ін'єкції на основі помилок, SQL-ін'єкції, засновані на логічних помилках, і SQL-ін'єкції на основі часу.

Ця вразливість присутня, якщо введені користувачем дані, які передаються в базовий SQL-вираз, можуть змінити значення оператора. Наприклад, наведений нижче код призначений для списку всіх користувачів з певним іменем (), яке було надано з HTML-форми userName

*statement = "SELECT * FROM users WHERE name = '" + userName + "';"*

Якщо користувач вкаже справжнє ім'я, інструкція працюватиме так, як задумано. Однак, зловмисник може повністю змінити поведінку цього SQL-виразу на новий оператор у наступному прикладі, вказавши для параметра *.a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 'tuserName*

*SELECT * FROM users WHERE name = 'a'; DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't';*

Модифікований оператор створює дійсний SQL-вираз, який видаляє таблицю та вибирає всі дані з таблиці (що розкриває інформацію кожного користувача). Це працює, тому що перша частина введеного тексту завершує вихідний *onepamop.usersuserinfoa'*;

Щоб уникнути такого роду атак, необхідно переконатися, що будь-які дані користувача, які передаються в SQL-запит, не можуть змінити характер запиту. Одним із способів зробити це є екранування всіх символів у введенні користувачем, які мають особливе значення в SQL.

SQL-вираз розглядає символ ' як початок і кінець рядкового літерала.

Розміщуючи зворотну похилу риску перед цим символом (\'), ми екрануємо символ і кажемо SQL замість цього обробляти його як символ (просто частину рядка).

У наступному операторі ми екрануємо символ '. Тепер SQL буде інтерпретувати ім'я як весь рядок, виділений жирним шрифтом (що насправді дуже дивне ім'я, але не шкідливе).

*SELECT * FROM users WHERE name = 'a\'; DROP TABLE users; SELECT * FROM userinfo WHERE \'t\' = \'t';*

Веб-фреймворки часто дбають про приховування персонажа. Django, наприклад, гарантує, що будь-які дані користувача, передані наборам запитів (запитам моделі), буде екрановано.

Підробка міжсайтових запитів (CSRF). CSRF-атаки дозволяють зловмиснику виконувати дії, використовуючи облікові дані іншого користувача без відома або згоди цього користувача [48-49].

Цей тип атаки найкраще пояснити на прикладі. Зловмисник, який знає, що певний сайт дозволяє користувачам, які увійшли в систему, надсилати гроші на вказаний обліковий запис за допомогою HTTP-запиту, який включає ім'я облікового запису та суму грошей. Зловмисник створює форму, яка містить його банківські реквізити та суму грошей у вигляді прихованих полів, і надсилає її електронною поштою іншим користувачам сайту (з кнопкою «Надіслати», замаскованою під посилання на сайт «швидко розбагатіти»).POST

Якщо користувач натисне кнопку «Надіслати», на сервер буде надіслано HTTP-запит, що містить деталі транзакції та будь-які клієнтські файли cookie, які браузер пов'язав із сайтом (додавання пов'язаних файлів cookie сайту до запитів є нормальною поведінкою браузера). Сервер перевірить файли cookie та використає їх, щоб визначити, чи увійшов користувач у систему та чи має дозвіл на здійснення транзакції.

Результатом є те, що будь-який користувач, який натисне кнопку «Надіслати», увійшовши на торговий сайт, здійснить транзакцію , зловмисник багатіє. Хитрість тут полягає в тому, що зловмиснику не потрібно мати доступ

до файлів cookie користувача (або облікових даних доступу). Браузер користувача зберігає цю інформацію і автоматично включає її в усі запити до пов'язаного сервера.

Один із способів запобігти цьому типу атак полягає в тому, щоб сервер вимагав, щоб запити включали секрет, створений для конкретного користувача. Секрет буде надано сервером під час надсилання веб-форми, яка використовується для здійснення переказів. Такий підхід заважає зловмиснику створити власну форму, адже йому довелося б знати секрет, який сервер надає користувачеві. Навіть якби він дізнався секрет і створив форму для конкретного користувача, він більше не зміг би використовувати ту саму форму для атаки на кожного користувача.

Інші поширені атаки/вразливості включають:

1) клікджекінг - під час цієї атаки зловмисник викрадає кліки, призначені для видимого сайту верхнього рівня, і спрямовує їх на приховану сторінку під ним. Ця техніка може бути використана, наприклад, для відображення законного банківського сайту, але захоплення облікових даних для входу в невидимий <*iframe*> контрольований зловмисником. Клікджекінг також можна використовувати, щоб змусити користувача натиснути кнопку на видимому сайті, але при цьому насправді мимоволі натиснути зовсім іншу кнопку. В якості захисту ваш сайт може запобігти вбудовуванню себе в *iframe* на іншому сайті, встановивши відповідні HTTP-заголовки.

2) відмова в обслуговуванні (DoS) - зазвичай досягається шляхом наповнення цільового сайту фальшивими запитами, щоб доступ до сайту порушувався для законних користувачів. Запитів може бути багато, або вони можуть окремо споживати великі обсяги ресурсів (наприклад, повільне читання або завантаження великих файлів). Захист від DoS-атак зазвичай працює, виявляючи та блокуючи «поганий» трафік, пропускаючи легітимні повідомлення. Ці засоби захисту, як правило, розташовані перед веб-сервером або в ньому (вони не є частиною самого веб-додатку).

3) обхід каталогів (Файл і розкриття інформації) - під час цієї атаки

зловмисник намагається отримати доступ до частин файлової системи веб-сервера, до яких він не повинен мати доступу. Ця вразливість виникає, коли користувач може передати імена файлів, які містять символи навігації файлової системи (наприклад,). Рішення полягає в тому, щоб продезінфікувати вхід перед його використанням.../../

4) включення файлів - у цій атаці користувач може вказати «ненавмисний» файл для відображення або виконання в даних, що передаються на сервер. При завантаженні цей файл може бути виконаний на веб-сервері або на стороні клієнта (що призведе до XSS-атаки). Рішення полягає в тому, щоб продезінфікувати вхід перед його використанням.

5) командна ін'єкція: атаки впровадження команд дозволяють зловмиснику виконувати довільні системні команди в операційній системі хоста. Рішення полягає в тому, щоб продезінфікувати введені користувачем дані, перш ніж їх можна буде використовувати в системних викликах.

2.2 Сучасні методи виявлення вразливостей веб-додатків

З поширенням Інтернету та появою безлічі Інтернет-підключених додатків, важливість безпеки додатків стає все більш критичною.

У 2022 році ін’єкції посідають третє місце за небезпеку після Broken Access Control and Cryptographic Failures. Згідно зі звітом Edgescan's Vulnerability Statistics за 2022 рік, у 2021 році було зламано понад 2511 мільярдів записів, забагато з цими пошкодженими, викликаними уразливістю рівня веб. -додатків, які могли бути уникають, якщо надійна безпека розвитку та видимість були дотримані [50]. Згідно зі звітом також, кількість високого ризику або критичної кількості вразливостей, виявлених у зовнішніх веб-додатках, зросла з 19,2% у 2018 році до 34,78% у 2019 році (рис.2.1).

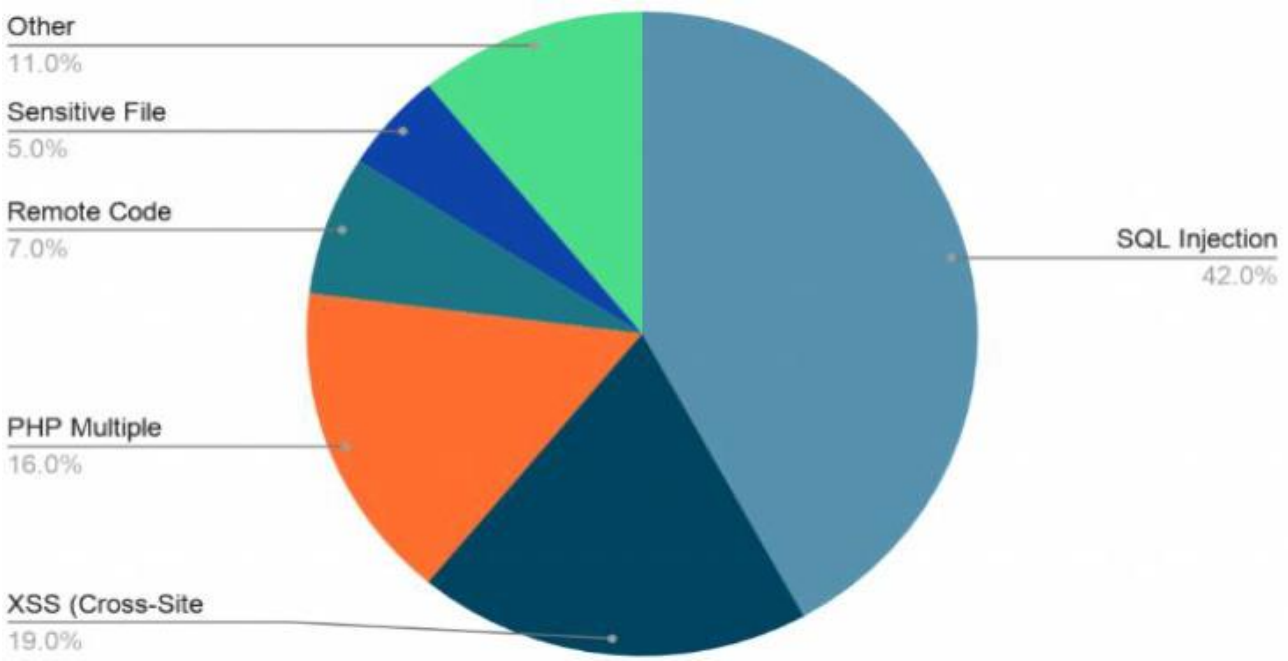

Рис.2.1. Співвідношення вразливостей веб-додатків

Крім того, число високих ризиків або серйозні вразливості, виявлені в системах мережевого рівня, значно зросли більше ніж удвічі в 2019.

В більшості наукових досліджень [51-52] визначені наступні методи виявлення вразливостей веб-додатків:

- автоматизовані сканери вразливостей - проходять по веб-додатку, виявляючи потенційні проблеми безпеки;
- пентестинг (тестування на проникнення) - експерти спробують вразити систему шляхом імітації атак та виявлення слабких місць;
- тестування навантаження та стресу - спроби визначити, як веб-додаток реагує на різні рівні трафіку та стресу;
- аналіз вихідного коду - використання інструментів статичного аналізу коду для виявлення потенційних вразливостей на етапі розробки;
- застосування технік реверс-інжинірингу - аналіз виконуваного коду для виявлення можливих слабких місць та вразливостей;
- аналіз трафіку - вивчення трафіку мережі та аналіз HTTP-запитів для виявлення аномалій та атак на рівні транспортного рівня;

- моніторинг безпеки в реальному часі - використання систем моніторингу безпеки для виявлення та запобігання атак у реальному часі;

- застосування інтерактивних інструментів для тестування віддалених служб - використання інструментів для тестування та валідації віддалених API на наявність вразливостей.

Уразливості можна знайти в додатках за допомогою різних методів. Вони можуть зберігатися в таємниці або повідомлятися постачальникам, публічно або приватно. Постачальники по-різному реагують на ці повідомлення та по-різному підходять до вирішення проблеми. Окрім виправлення відомих вразливостей, постачальники можуть вживати превентивних заходів, щоб уникнути вразливостей у першу чергу, або застосовувати методи глибокого захисту для пом'якшення наслідків. Ми детально розглянемо ці аспекти, починаючи з пошуку вразливостей і закінчуючи методами запобігання та пом'якшення наслідків.

Як видно з попередніх підрозділів роботи, уразливості можна знайти, уважно прочитавши вихідний код. Цей метод називається аудитом коду і зазвичай виконується навченими аудиторами безпеки. Аудитори безпеки можуть використовувати інструменти для допомоги. Ці інструменти виділяють місця вихідного коду, які потенційно містять вразливість. Однак помилкові спрацьовування зустрічаються досить часто. Повідомляється, що ці локації містять вразливість, хоча з ними все гаразд.

Крім того, існує багато помилкових негативних результатів, оскільки неможливо виявити всі вразливості автоматично.

По-перше, інструменти аудиту коду застосовують евристику, тобто наближення того, як може поводитися вихідний код; Вони настільки хороші, наскільки хороші їх евристики.

По-друге, повне міркування про вихідний код було б рівнозначно вирішенню проблеми зупинки, що, як відомо, неможливо. Тому повне міркування неможливе.

По-третє, виявлення логічних помилок, пов'язаних з безпекою, тобто

помилок, які є дуже специфічними для конкретної поведінки програми, вимагає машинозчитуваної специфікації поведінки програми, якої в більшості випадків не існує. Крім того, специфікація не обов'язково охоплює людський намір, а отже, сама по собі є помилковою. Отже, інструменти ніколи не зможуть замінити аудитора безпеки в аудиті коду.

Для проведення аудиту коду потрібен доступ до вихідного коду програми. Якщо застосунок не є програмним забезпеченням з відкритим вихідним кодом, вихідний код, як правило, недоступний для зовнішніх аудиторів, які аналізують програму без інструкцій постачальника. У цьому випадку аудитори повинні виконати реверс-інжиніринг, тобто зрозуміти машинний код програми, який призначений для запуску комп'ютером і нелегко зрозумілий людині. Навіть з підтримкою інструменту неможливо повністю відновити вихідний код. Незважаючи на ці перешкоди, багато вразливостей виявляються за допомогою методів зворотного інжинірингу.

Ще одна техніка – фаззінг. Фаззинг передає в програму мільйони різних випадкових вхідних даних і перевіряє наявність ненавмисної поведінки, наприклад збоїв. Збій є хорошим показником існування уразливості. Вхідні дані, які призводять до збоїв, потім зберігаються для подальшого аналізу. Щоб згенерувати ці вхідні дані, фаззер змінює існуючі вхідні дані та спостерігає, які частини програми виконуються з урахуванням змінених вхідних даних. Щоб збільшити ймовірність збою, фаззер намагається виконати всі частини програми. Мотивація цього методу полягає в тому, щоб знайти частини, які зазвичай не виконуються на очікуваних входах користувача і, отже, не перевірені на наявність помилок (безпеки). Фаззинг виявився напрочуд ефективним: наприклад, фаззер знайшов кілька вразливостей у популярному програмному забезпеченні OpenVPN навіть після того, як вже було проведено два аудити коду.

Після того, як вразливість буде знайдена, аудитор безпеки може вирішити зберегти її в таємниці або повідомити про неї. Мотиви для збереження вразливості в таємниці включають сплановані злочинні дії, шпигунство з боку спецслужб і доступ правоохоронних органів до пристрою підозрюваного. У всіх

цих випадках, швидше за все, розробляється експлойт, щоб скористатися вразливістю. Постачальник не може виправити вразливість, якщо він не знає про неї. Таким чином, незареєстровані вразливості часто залишаються невиправленими протягом тривалого часу. Уразливості без виправлення називаються «нульовими днями» або «0-днями».

Існує два підходи до публікації вразливостей: повне розкриття та відповідальне розкриття. При повному розкритті уразливість розкривається публічно, без попереднього повідомлення постачальника. Прихильники повного розкриття інформації стверджують, що всі користувачі вразливого програмного забезпечення повинні мати однакову інформацію про вразливість, щоб мати можливість оцінити свої ризики та вжити відповідних контрзаходів, доки не буде випущено виправлення. Вони приймають ризик того, що зловмисники можуть використовувати інформацію для розробки експлойту та націлювання на користувачів вразливого програмного забезпечення. Крім того, прихильники повного розкриття інформації стверджують, що повне розкриття інформації чинить більший тиск на постачальника, щоб він швидше створював і відправляв виправлення, а також більше дбав про безпеку в першу чергу.

На відміну від повного розкриття інформації, відповідальне розкриття інформації (іноді його також називають скоординованим розкриттям інформації) вимагає спочатку повідомити постачальника, зазвичай надаючи йому конкретні часові рамки для випуску виправлення перед виходом на біржу. Тривалість цього ембарго є компромісом між тиском на постачальника та наданням йому можливості ретельно дослідити проблему, включаючи ретельне тестування виправлення. Типове значення – 90 днів. Продавці можуть просити про продовження ембарго. Однак це залишається на розсуд того, хто його надає. Наприклад, був резонансний випадок, коли дослідники безпеки, які працюють у Google, не надали Microsoft продовження.

2.3 Сучасні методи аналізу та оцінювання вразливостей веб-додатків

Згідно зі статистичним звітом WhiteHat про безпеку додатків за 2017 рік, 30% від загальної кількості зареєстрованих порушень стосувалися атак на веб-додатки [53] . Це привід для компаній задуматися, наскільки безпечними є їхні програми. Щоб оцінити безпеку веб-додатків, компанії звертаються до постачальників послуг з оцінки безпеки. Провайдери пропонують дві основні методики: перевірку вихідного коду та тестування на проникнення (рис.2.2).

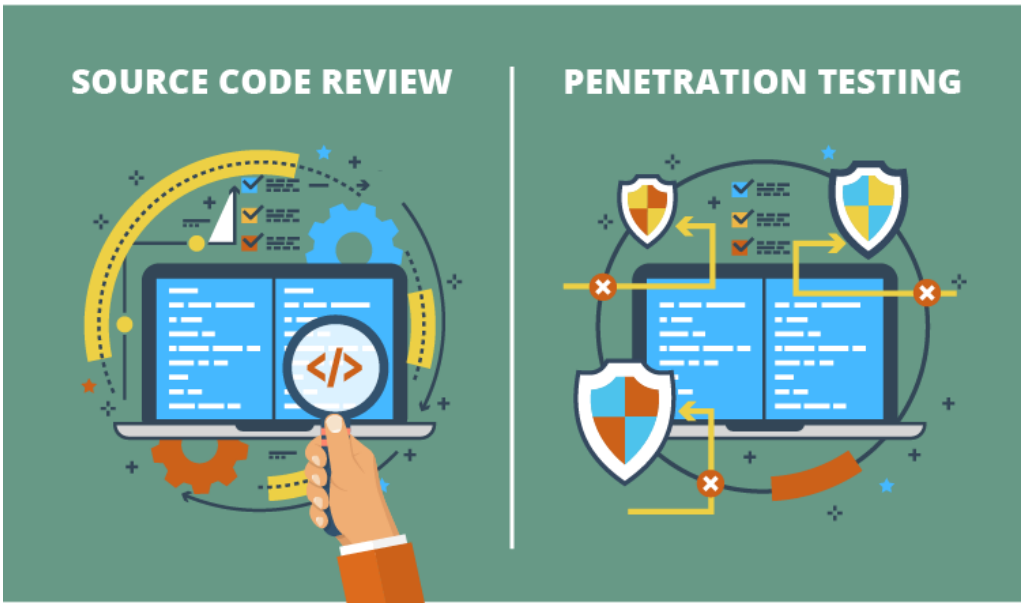

Рис. 2.2. Методики перевірки веб-додатків

Аналіз безпеки вихідного коду (source code review) – це перевірка вихідного коду програми для пошуку помилок, пропущених на початковому етапі розробки. Тестувальник запускає аналізатор коду, який сканує порядковий код програми. Після того, як аналізатор, розгорнутий у тестовому середовищі, знаходить вразливості, пентестер вручну перевіряє їх, щоб виключити помилкові спрацьовування.

Час, який тестувальник витрачає на перевірку вихідного коду, залежить від мови програмування та розміру програми. Наприклад, аналіз 1000 рядків коду може зайняти 0,5 – 2 години.

Сучасні методи аналізу та оцінювання вразливостей веб-додатків охоплюють різноманітні підходи та інструменти для планування безпекових заходів та реагування на потенційні загрози. Деякі методи аналізу та оцінювання вразливостей веб-додатків включають [54,55]:

- ручна перевірка безпеки:
- експерти вручну перевіряють веб-додаток на наявність вразливостей, які можуть залишитися непоміченими автоматизованими інструментами.
- аналіз вихідного коду: вивчення вихідного коду на предмет вразливостей та виконання докладного аналізу;
- пентестинг (тестування на проникнення): експерти оцінюють вразливості та ризики, які можуть виникнути внаслідок потенційних атак;
- застосування інтерактивних інструментів для тестування віддалених служб: тестування і валідація віддалених служб та API на предмет вразливостей;
- моніторинг безпеки в реальному часі: аналіз подій та сповіщень систем моніторингу для визначення серйозності та обсягу вразливостей.

Ці методи використовуються в комбінації для максимальної ефективності та повноти виявлення та оцінювання вразливостей веб-додатків.

З появою нових моделей розробки та розгортання з’являються нові інструменти тестування для їх захисту [56-60]:

- сканування вразливостей безпеки API (ASV) для виявлення та зменшення ризику, пов’язаного з використанням API, включаючи безсерверні API;
- інструменти безпеки інфраструктури як коду для захисту від неправильних конфігурацій, порушень політики, загроз і викликів IAM;
- SAST – інструменти статичного тестування безпеки додатків (SAST) для виявлення та усунення вразливостей у вихідному коді вашої організації;
- SCA – інструменти тестування аналізу складу програмного забезпечення (SCA) для виявлення використання компонентів із відкритим кодом і пов’язаних з ними вразливостей;
- IAST – інструменти інтерактивного тестування безпеки додатків (IAST)

для виявлення та усунення ризиків під час роботи в веб-додатках вашої організації;

- DAST - інструменти динамічного тестування безпеки додатків (DAST) для виявлення та усунення ризиків у додатках;
- плагіни IDE активно та успішно використовуються для допомоги у виявленні та вирішенні проблем безпеки;
- інструменти сканування мікросервісів для виявлення використання компонентів і виявлення вразливостей у базових зображеннях і конкретних артефактах;
- фаззинг для виявлення проблем безпеки та стабільності;
- інструменти безпеки конфігурації середовища виконання контейнера для забезпечення безпечної конфігурації.

Однак, за результатами опитування організацій щодо застосування інструментів виявлення вразливостей не всі їх повністю застосовують (рис.2.3).

Рис. 2.3. Застосування інструментів сканування безпеки веб-додатків

Сильною стороною перевірки вихідного коду є можливість виявити такі вразливості:

- помилки шифрування, до них відносяться слабкі алгоритми шифрування, а також сильні алгоритми шифрування зі слабкою реалізацією (наприклад, небезпечне зберігання ключів);
- всі випадки SQL-ін'єкцій, вразливостей XSS (cross-site scripting);
- переповнення буфера (у буфер поміщається більше даних, ніж він може обробити);
- умови змагання (виконання двох і більше операцій одночасно).

Крім того, якщо тестування на проникнення дозволяє виявити вразливу веб-сторінку, то перевірка вихідного коду дозволяє пентестерам знаходити уразливості на кореневому рівні (для виявлення помилок у функції або модулі, що використовується на декількох веб-сторінках). Це економить час пентестера та гроші клієнта.

Тестування на проникнення – це процедура, під час якої пентестер зламує веб-додаток, щоб виявити вразливості в додатку. Процес більш трудомісткий, ніж перевірка вихідного коду, оскільки включає в себе кілька етапів. Спочатку пентестер проводить розвідку проти цільової програми за допомогою набору користувацьких тестів і запускає веб-сканер для пошуку точок входу. Після цього він або вона використовує вразливості, намагаючись підвищити привілеї до адміністративного рівня.

Залежно від складності веб-додатку, процедура може зайняти від 20 до 400 годин.

Використання різниться, як і інструменти, які організації вважають найважливішими, але більшість організацій зрештою повинні використовувати комплекс інструментів для задоволення своїх потреб у безпеці.

Деякі вразливості можна виявити лише за допомогою тестування на проникнення:

- індексація пошукових систем: місцеві пошукові системи додатка можуть розкривати конфіденційні дані пентестерів, такі як копії паспорта або

водійського посвідчення;

- уразливості, спричинені неправильною конфігурацією - це випадки, коли внутрішні служби та документи доступні через Інтернет, використання облікових даних за замовчуванням, таких як «admin» та «user»;

- слабка автентифікація, наприклад, слабкий пароль або CAPTCHA, повторне використання пароля;

- логічні помилки в рольовому доступі, коли певна інформація доступна більш широкому колу користувачів.

Крім того, тестування на проникнення вимагається стандартами безпеки. Наприклад, відповідність Закону про мобільність і підзвітність медичного страхування (HIPAA) у США включає двофакторну автентифікацію, автоматичний вихід із системи та екстрений доступ до електронної захищеної медичної інформації (EPHI).

Ключова перевага тестування на проникнення полягає в тому, що воно базується на оцінці ризиків. На етапі розвідки пентестер дізнається про бізнес замовника через веб-додаток. Це допомагає виявляти високопріоритетні ризики та створювати тест-кейси для конкретного бізнесу. Наприклад, якщо цільовою програмою є веб-сайт локальної пошукової системи, пентестер надаватиме пріоритет вразливостям, які призводять до атак інтелектуального аналізу даних, над вразливостями XSS.

Перевірка вихідного коду перевіряє якість коду веб-додатку. Тестування на проникнення, в свою чергу, виявляє проблеми з логікою веб-додатків. Перевірка вихідного коду + тестування на проникнення, проведене різними пентестерами, є ефективною комбінацією, яка покриває більшість вразливостей веб-додатків.

У випадку з корпоративними веб-додатками доцільніше інвестувати в кіберзахист, ніж усувати порушення безпеки. Таким чином, витрати на перевірку коду та тестування на проникнення завжди окупаються.

Існують інструменти на сайті HackYourMom для аналізу веб-додатків (рис 2.4.).

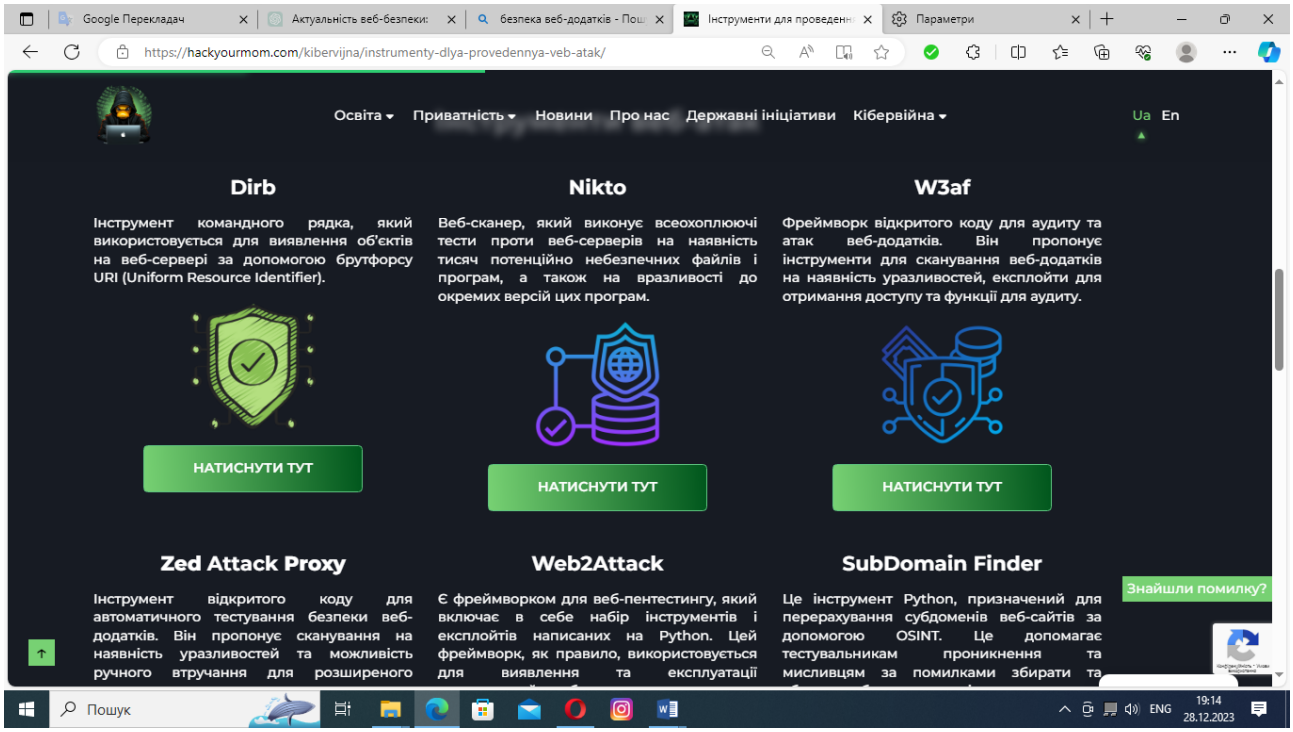

Рис. 2.4. Сайт з набором інструментів для тестування веб-додатків

Висновок до другого розділу. Традиційні методи аналізу та оцінювання вразливостей веб-додатків вимагають великої затрати часу на проведення роботи. Визначення типових вразливостей веб-додатків є критично важливим етапом аналізу безпеки. У цьому підрозділі проведено аналіз різноманітних вразливостей, таких як SQL-ін'єкції, перехоплення сесій, вразливості введення та інші. Цей аналіз надає зрозуміння основних точок атак та дозволяє спрямовувати увагу на найбільш критичні аспекти безпеки веб-додатків. Зроблений аналіз дає можливість вибрати оптимальні методи для конкретних завдань забезпечення безпеки веб-додатків.

Розглянуті такі аспекти, як статичний та динамічний аналіз коду, оцінка безпеки архітектури, аудит конфігурації та використання інструментів для оцінки надійності заходів безпеки створюють основу для пошуку нових методик комплексної роботи в управлінні кібербезпекою з використанням сучасних підходів до створення системи виявлення, аналізу та оцінювання вразливостей веб-додатків.

Встановлено, що одним із шляхів є автоматизація виявлення, аналізу та

оцінювання вразливостей, створення комплексних методик на основі застосування штучного інтелекту, нейронних мереж та машинного навчання, на які потрібно зосередити увагу в подальших дослідженнях.

Визначено, що забезпечення безпеки веб-додатків - це постійний процес, і заходи забезпечення безпеки повинні оновлюватися та адаптуватися до нових загроз та технічних вимог.

РОЗДІЛ 3
РОЗРОБКА КОМПЛЕКСНОЇ МЕТОДИКИ ВИЯВЛЕННЯ, АНАЛІЗУ ТА ОЦІНЮВАННЯ ВРАЗЛИВОСТЕЙ СУЧАСНИХ ВЕБ-ДОДАТКІВ

3.1 Комплексний підхід до виявлення та оцінювання вразливостей веб-ресурсів

Комплексний підхід до виявлення та оцінювання вразливостей веб-додатків включає в себе комбінацію автоматизованих і ручних методів, щоб забезпечити повноту та надійність процесу безпеки.

Нижче представлено алгоритм для комплексного підходу (рис.3.1):

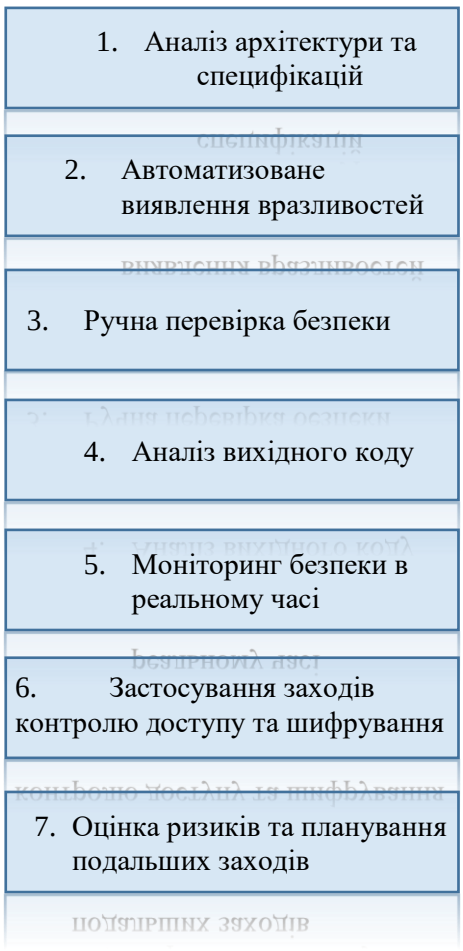

Рис. 3.1. Алгоритм комплексного підходу до виявлення та оцінювання вразливостей веб-додатків

Крок 1: Аналіз архітектури та специфікацій. Збір інформації: отримання повної інформації про веб-додаток, його функції, архітектуру та залежності.

Аналіз коду: вивчення вихідного коду для розуміння структури та можливих вразливостей на ранніх етапах.

Крок 2: Автоматизоване виявлення вразливостей. Запуск автоматизованих сканерів: Використання інструментів, таких як Burp Suite, OWASP ZAP, Nessus, для сканування веб-додатка та виявлення різних вразливостей.

Тестування вантажу та стресу: використання інструментів для тестування ефективності та стійкості веб-додатка під навантаженням.

Крок 3: Ручна перевірка безпеки. Проведення пентесту: експерти проводять тестування на проникнення, спробуючи визначити вразливості, які можуть бути пропущені автоматизованими інструментами.

Ручна перевірка бізнес-логіки: перевірка коректності виконання бізнес-логіки веб-додатка та ідентифікація можливих аномалій.

Крок 4: Аналіз вихідного коду. Використання статичних аналізаторів коду: використання інструментів, таких як SonarQube або Fortify, для оцінки вихідного коду на предмет вразливостей та дотримання стандартів безпеки.

Крок 5: Моніторинг безпеки в реальному часі. Встановлення систем моніторингу безпеки: використання Web Application Firewalls (WAF) та систем моніторингу подій для виявлення та реагування на вразливості в реальному часі.

Крок 6: Застосування заходів контролю доступу та шифрування. Контроль доступу: встановлення правильних заходів контролю доступу, щоб обмежити права користувачів та забезпечити принцип найменших привілеїв.

Шифрування даних: використання шифрування для захисту конфіденційних даних під час передачі та зберігання.

Крок 7: Оцінка ризиків та планування подальших заходів. Оцінка ризиків: визначення потенційних наслідків та ймовірності виникнення вразливостей.

Розробка стратегії покращення безпеки: планування та впровадження подальших заходів для покращення безпеки веб-додатка.

3.2 Застосування штучного інтелекту для підвищення ефективності безпеки веб-додатків

Одним із шляхів підвищення ефективності безпеки веб-додатків є застосування у визначному алгоритмі інструментів штучного інтелекту.

Застосування штучного інтелекту (ШІ) для підвищення ефективності веб-додатків може включати різноманітні аспекти, від безпеки та аналізу до покращення користувацького досвіду. Дослідники у своїх роботах [61-63] визначають кілька способів, як ШІ може бути використаний для підвищення ефективності веб-додатків:

1. Забезпечення безпеки:

а) виявлення вразливостей: використання алгоритмів машинного навчання для виявлення потенційних вразливостей в коді або конфігураціях веб-додатка;

б) аналіз трафіку: моніторинг та аналіз мережевого трафіку для виявлення аномалій, що можуть свідчити про атаки;

в) системи виявлення та запобігання інтранет-атак (IDPS): використання ШІ для розпізнавання та блокування потенційно шкідливого трафіку в реальному часі.

2. Оптимізація використання ресурсів:

а) масштабування ресурсів: використання алгоритмів ШІ для автоматизованого масштабування ресурсів сервера в залежності від навантаження;

б) оптимізація продуктивності коду: використання аналізу коду на основі ШІ для ідентифікації слабких місць та вдосконалення продуктивності.

3. Покращення користувацького досвіду:

а) персоналізація контенту: використання алгоритмів рекомендацій для персоналізації контенту та рекомендацій для користувачів;

б) чат-боти та віртуальні асистенти: використання ШІ для створення інтелектуальних чат-ботів або віртуальних асистентів для підтримки

користувачів.

4. Аналіз та Прогнозування:

а) аналіз Даних користувачів: використання ШІ для аналізу поведінки користувачів та розвитку стратегій покращення користувацького досвіду;

б) прогнозування витрат ресурсів: використання алгоритмів ШІ для прогнозування витрат ресурсів та оптимізації їх використання.

5. Робота з контентом:

а) генерація контенту: використання генеративних моделей ШІ для автоматизованої генерації текстового або мультимедійного контенту;

б) класифікація та фільтрація: використання алгоритмів для класифікації та фільтрації контенту з метою забезпечення безпеки та якості.

6. Автоматизоване Тестування:

а) тестування вразливостей: використання ШІ для автоматизованого виявлення вразливостей в коді та конфігураціях;

б) автоматизовані тести відповіді на стрес використання ШІ для автоматизації тестування відповіді веб-додатка на стрес та перевантаження.

Загальний підхід полягає в інтеграції різних модулів ШІ в різні аспекти розробки та експлуатації веб-додатків для покращення безпеки, продуктивності та користувацького досвіду.

Наприклад, одним із запропонованих варіантів застосування штучного інтелекту є його використання для підтримки користувачів у сфері безпеки веб-додатків. Створення інтелектуального чат-бота або віртуального асистента для підтримки користувачів у сфері безпеки веб-додатків може бути корисним для надання інформації, допомоги з питань безпеки та відповідей на питання. Давайте розглянемо кроки для створення такого чат-бота з використанням Python та бібліотек для обробки природної мови (NLP) та веб-фреймворків.

Створення інтелектуального чат-бота або віртуального асистента для підтримки користувачів у сфері безпеки веб-додатків може бути корисним для надання інформації, допомоги з питань безпеки та відповідей на питання. Давайте розглянемо кроки для створення такого чат-бота з використанням

Python та бібліотек для обробки природної мови (NLP) та веб-фреймворків.

1. Вибір Бібліотеки NLP для обробки природної мови для розпізнавання та розуміння тексту, який вводить користувач. Це може бути:

spaCy: високоефективна бібліотека для обробки природної мови, яка може визначати сутності, розбивати текст на токени, проводити аналіз залежностей та інше;

NLTK (Natural Language Toolkit) - широко використовувана бібліотека для роботи з текстом та обробки природної мови. Вона включає в себе різноманітні інструменти для токенізації, лематизації, частиномовного аналізу та інше.

2. Вибір Веб-фреймворку для реалізації веб-інтерфейсу для чат-бота (рис. 3.2): Flask - легкий веб-фреймворк для Python, який дозволяє швидко створювати веб-застосунки; Django - повноцінний веб-фреймворк для Python, який надає багато вбудованих функцій та можливостей.

```
from flask import Flask, render_template, request
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer

app = Flask(__name__)

# Створення чат-бота
chatbot = ChatBot('SecurityBot')
trainer = ChatterBotCorpusTrainer(chatbot)
trainer.train('chatterbot.corpus.english')

# Веб-інтерфейс
@app.route("/")
def home():
    return render_template("index.html")

# Обробка запитань користувача
@app.route("/get")
def get_bot_response():
    user_text = request.args.get('msg')
    return str(chatbot.get_response(user_text))

if __name__ == "__main__":
    app.run(debug=True)
```

Рис.3.2. Лістинг коду на Python для створення простого веб-чат-бота

Спочатку ми створюємо чат-бота за допомогою ChatterBot. Ми навчаємо його, використовуючи вбудовані корпуси англійської та української мови, що дозволяє також врахувати особливості спілкування фахівців кібербезпеки, яке відрізняється установленими сленгами:

```
chatbot = ChatBot('SecurityBot')
trainer = ChatterBotCorpusTrainer(chatbot)
trainer.train('chatterbot.corpus.english') //'path/to/ukrainian_corpus.yml'
```

Створення Веб-інтерфейсу з Flask:

Далі ми використовуємо Flask для створення веб-інтерфейсу. Ми створюємо три шляхи: один для домашньої сторінки та два для обробки запитань користувача.

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")

@app.route("/get")
def get_bot_response():
    user_text = request.args.get('msg')
    return str(chatbot.get_response(user_text))

if __name__ == "__main__":
    app.run(debug=True)
```

І нарешті, в папці з вашим додатком Flask маємо папку templates, в якій ми зберігаємо HTML-шаблон для веб-сторінки. Наприклад, index.html має вигляд:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SecurityBot</title>
</head>
<body>
    <h1>SecurityBot</h1>
    <div id="chatbox">
        <div class="chat" id="chat"></div>
        <input type="text" id="userInput" placeholder="Питання...">
        <button onclick="sendMessage()">Надіслати</button>
    </div>
    <script>
        function sendMessage() {
            var userMessage = document.getElementById("userInput").value;
            document.getElementById("chat").innerHTML += "<p><strong>Ви:</strong> " +
userMessage + "</p>";
            document.getElementById("userInput").value = "";
            // Відправка запиту до сервера
            var xhr = new XMLHttpRequest();
            xhr.open("GET", "/get?msg=" + userMessage, true);
            xhr.onreadystatechange = function() {
                if (xhr.readyState == 4 && xhr.status == 200) {
                    var botMessage = xhr.responseText;
                    document.getElementById("chat").innerHTML                    +=
"<p><strong>SecurityBot:</strong> " + botMessage + "</p>";
                }
            };
            xhr.send();
        }
    </script>
</body>
</html>
```

Його зовнішній вигляд за кілька хвилин (рис.3.3):

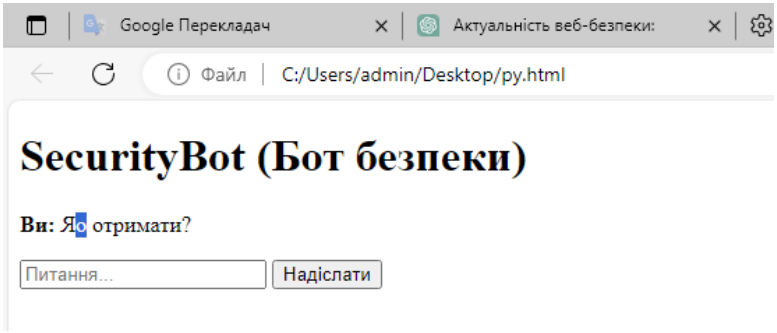

Рис.3.3. Інтерфейс створеного бота безпеки

Це лише простий приклад, який можна розширити його функціональність та вдосконалити відповідно до потреб та сценаріїв безпеки веб-додатків.

3. Використання Інструментів NLP та Чат-бота для реалізації чат-бота, такі як ChatterBot чи Rasa, які можуть навчатися та розуміти користувацькі запитання:

ChatterBot - простий інструмент для створення чат-ботів, який може навчатися на основі корпусів текстів;

Rasa - можливо найбільш потужний відкритий фреймворк для розробки чат-ботів. Він надає інструменти для розпізнавання інтентів, обробки діалогів та багато іншого.

3.3 Комплексна методика виявлення, аналізу та оцінювання вразливостей веб-додатків

Існують комплексні методики виявлення, аналізу та оцінювання вразливостей веб-додатків, які охоплюють різні аспекти безпеки та включають в себе як автоматизовані інструменти, так і ручні процеси. Однією з таких методик є "OWASP Application Security Verification Standard" (ASVS) та "OWASP Testing Guide" [29].

1. OWASP Application Security Verification Standard (ASVS):

OWASP ASVS визначає стандартні вимоги безпеки для розроблення та оцінки веб-додатків та служб. ASVS складається з трьох рівнів (1, 2 та 3), призначених для використання в різних сценаріях відповідно до складності та чутливості додатка (рис.3.4).

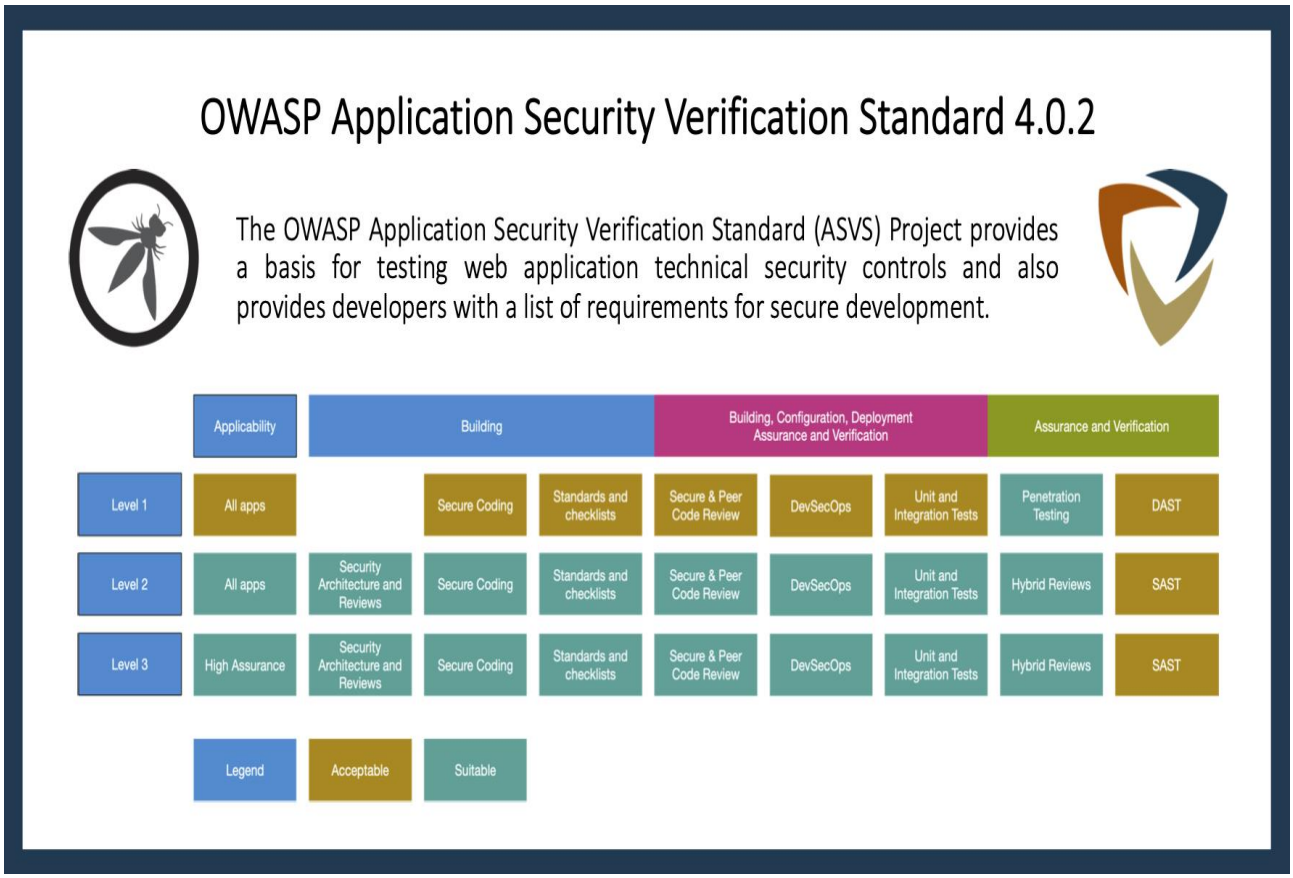

Рис. 3.4. Три рівні вимог безпеки OWASP ASVS

Кожен рівень ASVS має набір вимог безпеки, які охоплюють різні аспекти, такі як ідентифікація та автентифікація, управління сесіями, контроль доступу, захист від зламу, захист даних та інші. ASVS може використовуватися для виявлення та оцінювання вразливостей веб-додатків відповідно до стандартів безпеки.

Основною метою проекту OWASP Application Security Verification Standard (ASVS) є нормалізація діапазону охоплення та рівня строгості, доступних на ринку, коли справа доходить до виконання перевірки безпеки веб-додатків за допомогою комерційно працездатного відкритого стандарту. Стандарт

забезпечує основу для тестування технічних засобів контролю безпеки додатків, а також будь-яких технічних засобів контролю безпеки в середовищі, які покладаються для захисту від вразливостей, таких як міжсайтовий скриптинг (XSS) і SQL-ін'єкція. Цей стандарт може бути використаний для встановлення рівня впевненості в безпеці веб-додатків. Вимоги були розроблені з урахуванням наступних цілей:

Використання в якості метрики – для надання розробникам додатків і власникам додатків мірила, за допомогою якого можна оцінити ступінь довіри, який може бути наданий їх веб-додаткам,

Використовувати як керівництво - надавати вказівки розробникам засобів контролю безпеки щодо того, що потрібно вбудовувати в елементи керування безпекою, щоб задовольнити вимоги безпеки програм;

Використання під час закупівель - є підставою для зазначення в договорах вимог до перевірки безпеки додатків.

2. OWASP Testing Guide (рис.3.5):

Рис.3.5. Посібник застосування OWASP Testing Guide

OWASP Testing Guide - це великий проект, який містить матеріали, призначені для використання при проведенні тестів на безпеку веб-додатків та

служб. Він включає в себе рекомендації щодо проведення тестів на безпеку на різних етапах життєвого циклу розробки, від визначення вимог до впровадження та експлуатації.

OWASP Testing Guide включає різноманітні теми, такі як аналіз вхідних даних, тестування автентифікації та авторизації, тестування безпеки API, тестування для розробників, тестування мобільних додатків та інше. Це допомагає розробникам та тестувальникам виявляти та виправляти вразливості веб-додатків.

Ці методики, разом із стандартами безпеки, такими як ISO/IEC 27001 та NIST SP 800-53 [32,34] надають комплексний підхід до забезпечення безпеки веб-додатків на різних етапах їх життєвого циклу.

Висновок до розділу 3. Вивчення і розробка комплексного підходу – це крок до покращення ефективності виявлення та оцінювання вразливостей. У цьому розділі був розроблений та описаний комплексний метод, який враховує різноманітні аспекти виявлення, аналізу та оцінювання вразливостей на кожному етапі життєвого циклу веб-додатка. Розглянуті інтегровані стратегії, спрямовані на виявлення не тільки вже існуючих, але й потенційних вразливостей.

Використання штучного інтелекту є ключовим компонентом удосконалення систем безпеки веб-додатків. У розділі досліджено та методи та алгоритми штучного інтелекту для автоматизації процесів виявлення вразливостей, а також для вдосконалення аналізу та оцінювання їх серйозності.

Запропонований алгоритм, який складає основу нової комплексної методики виявлення, аналізу та оцінювання вразливостей сучасних веб-додатків.

Комплексну методику виявлення, аналізу та оцінювання вразливостей сучасних веб-додатків, враховуючи аналіз технологій виявлення та сучасні методи і вимоги міжнародних стандартів, доцільно поділити на чотири етапи.

По-перше, аналіз попередніх досліджень, пов’язаних із методами та інструментами для виявлення та запобігання вразливостям системи безпеки.

По-друге, розробляючи нову пропоновану систему для виявлення та

запобігання вразливостям, система визначає багато вразливостей веб-сайтів, як-от атаки міжсайтових сценаріїв, ін’єкцію SQL, віддалене виконання коду та сканування відбитків пальців.

По-третє, розробка запропонованої системи.

По-четверте, перевірка запропонованої системи шляхом практичного застосування на веб-сайті та демонстрація результатів. Загалом, запропонована методологія сприяє досягненню цілей дослідження, забезпечуючи систематичний підхід до виявлення та запобігання вразливостям безпеки у веб-додатках.

Поєднуючи огляд літератури, проведений аналіз, можна ствердити, що такий підхід до створення є ефективним, надійним та досягає поставлених цілей. Застосування штучного інтелекту підсилює ефективність методики та робить процес забезпечення безпеки більш автоматизованим та адаптованим до сучасних викликів.

РОЗДІЛ 4
ПРОПОЗИЦІЇ ТА РЕКОМЕНДАЦІЇ УДОСКОНАЛЕННЯ БЕЗПЕКИ ВЕБ-ДОДАТКІВ

4.1 Оцінка ефективності запропонованих стратегій

Оцінка ефективності безпеки веб-додатків та розроблених стратегій виявлення і аналізу вразливостей - це важливий етап, щоб переконатися, що додаток захищений від потенційних загроз та вразливостей. Нижче подано деякі ключові аспекти, які можна врахувати при оцінці ефективності безпеки веб-додатків:

1. Відповідність до стандартів та норм - перевірте, чи додаток відповідає стандартам безпеки, таким як OWASP ASVS, або нормативним документам держави [36-37], які можуть стосуватися вашої області діяльності.

2. Регулярне тестування на проникнення (Penetration Testing) - проведення регулярних тестів на проникнення для виявлення слабких місць та вразливостей.

3. Моніторинг та виявлення подій - встановлення систем моніторингу безпеки для виявлення непередбачених подій та аномалій.

4. Аудит коду та аналіз вразливостей - регулярне проведення аудиту вихідного коду та аналіз вразливостей для виявлення та виправлення проблем безпеки.

5. Планування та тестування бізнес-процесів - визначення та тестування сценаріїв відновлення додатку після атаки або випадку виникнення вразливостей.

6. Тестування навантаження та відновлення - тестування здатності додатка витримувати стресові навантаження та відновлюватися після непередбачених випадків.

7. Ефективність управління доступом - перевірка, наскільки ефективно реалізовано управління доступом та чи відповідає це бізнес-потребам.

8. Навчання безпеки для персоналу - забезпечення, розуміння персоналом загрози безпеки та вміння правильно реагувати на них.

9. Використання зовнішніх аудиторів та експертів - залучення зовнішніх аудиторів та експертів для незалежного оцінювання безпеки.

10. Постійне оновлення та вдосконалення - впровадження системи постійного оновлення та вдосконалення безпеки відповідно до нових загроз та технологічних тенденцій.

11. Безпекова архітектура та дизайн - оцінка та підтримка безпекової архітектури та дизайну веб-додатка.

12. Документація та спілкування - забезпечення належної документації безпеки та спілкування із зацікавленими сторонами (стейкхолдерами).

13. Забезпечення контролю доступу та захисту даних - реалізація ефективного контролю доступу та захисту конфіденційної інформації.

14. Оцінка правової відповідальності - перевірка, як добре додаток відповідає вимогам законодавства про захист персональних даних та інших правових норм.

Загальний підхід передбачає комплексну систему заходів та вимірювань для забезпечення ефективності безпеки веб-додатків на різних рівнях та етапах їхнього життєвого циклу.

4.2 Пропозиції щодо практичного застосування комплексної методики виявлення, аналізу та оцінювання вразливостей веб-додатків

Важливо наголосити, що самостійне тестування безпеки свого сервера повинно бути виконане обачливо та з відповідним знанням, оскільки неправильні дії можуть вплинути на його функціонування або викликати проблеми з безпекою.

Щоб визначити ефективність безпеки веб-сервера, можна використовувати різні підходи та інструменти. Ось деякі загальні поради та приклади коду на Python для тестування ефективності безпеки:

1. Сканування Вразливостей:

Використовуйте інструменти, такі як OWASP ZAP чи Nikto, для автоматизованого сканування вашого сервера на вразливості.

```
python
# Приклад використання Nikto через Python
import subprocess

def run_nikto(host):
    command = f"nikto -h {host}"
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    output, error = process.communicate()
    print(output.decode())

# Виклик функції з вказанням IP-адреси сервера
run_nikto("127.0.0.1")
```

Рис. 4.1. Лістинг програми застосування бібліотеки Nikto

2. Тестування на Проникнення:

Проведіть тестування на проникнення, використовуючи зразки атак, щоб переконатися, що ваш сервер стійкий до атак.

```
python
# Приклад використання sqlmap через Python для тестування SQL-ін'єкцій
import subprocess

def test_sql_injection(url):
    command = f"sqlmap -u {url} --level=5"
    process = subprocess.Popen(command, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
    output, error = process.communicate()
    print(output.decode())

# Виклик функції з вказанням URL
test_sql_injection("http://example.com/page?id=1")
```

Рис.4.2. Лістинг програми для тестування SQL-ін'єкцій

3. Моніторинг журналів та інцидентів:

Перевіряйте журнали подій та інцидентів для виявлення аномалій чи потенційно шкідливих дій.

```
python
# Приклад виводу журналу подій (логів)
import os
import datetime

def read_logs():
    log_file = "/var/log/apache2/error.log"  # Приклад шляху до журналу помилок
Apache
    with open(log_file, "r") as file:
        logs = file.read()
        print(logs)

# Виклик функції для читання журналу
read_logs()
```

Рис.4.3. Лістинг програми автоматизації аналізу журналів подій

Ці приклади коду дають загальну ідею про те, як можна автоматизувати деякі етапи тестування безпеки. Однак це лише частковий підхід, і для комплексного тестування ефективності безпеки веб-сервера рекомендується використовувати повноцінні інструменти та сервіси, які спеціалізуються на цьому завданні. Наприклад, для автоматизованого тестування на проникнення можна використовувати Metasploit чи Burp Suite (рис. 4.4, 4.5).

```
from metasploit.msfrpc import MsfRpcClient

# З'єднання з Metasploit RPC сервером
client = MsfRpcClient('your_metasploit_rpc_host', port=55553, token='your_rpc_token')

# Створення нової сесії метерпретатора
exploit = client.modules.use('exploit', 'multi/handler')
exploit['PAYLOAD'] = 'windows/meterpreter/reverse_tcp'
exploit['LHOST'] = 'your_local_ip'
exploit['LPORT'] = 4444

# Запуск експлоітації
exploit.execute()
# Чекання на з'єднання від цілі
```

Рис. 4.4. Приклад лістингу для використання Metasploit Framework на Python

```
from burp import IBurpExtender

# Приклад розширення для Burp Suite
class BurpExtender(IBurpExtender):
    def registerExtenderCallbacks(self, callbacks):
        self._callbacks = callbacks
        self._helpers = callbacks.getHelpers()
        callbacks.setExtensionName("My Burp Extension")

    def processHttpMessage(self, toolFlag, messageIsRequest, messageInfo):
        # Логіка обробки HTTP повідомлення
        pass
# Далі можна використовувати сесії Metasploit для виконання різних команд
```

Рис.4.5. Приклад лістингу для використання Burp Suite API на Python

При цьому важливо пам’ятати, що Metasploit та Burp Suite - це потужні інструменти для тестування на проникнення, але їх використання відбувається за умови легітимного, законного та етичного використання. Проведення тестування на проникнення без належної авторизації може порушувати закони та етичні норми.

Для виведення результатів сканування журналів та тестування на проникнення на Python графічно ви можете використовувати бібліотеки для візуалізації даних, такі як Matplotlib, Seaborn, Plotly, чи інші (рис. 4.6).

```
import matplotlib.pyplot as plt

# Припустимі дані (замініть це реальними результатами вашого тестування)
vulnerabilities = ['SQL Injection', 'Cross-Site Scripting', 'Insecure Configuration']
severity_levels = [5, 8, 3]

# Побудова графіку
plt.bar(vulnerabilities, severity_levels, color='blue')
plt.xlabel('Вразливості')
plt.ylabel('Рівень серйозності')
plt.title('Результати тестування на проникнення')
plt.show()
```

Рис. 4.6. Приклад лістингу для побудови графіків на основі отриманих даних

Цей приклад будує стовпчастий графік, де на осі X відображаються типи вразливостей, а на осі Y - рівень серйозності. Конкретній організації слід адаптувати цей код для використання конкретних даних та результатів тестування.

SQL-ін'єкціям можна запобігти, використовуючи підготовлені оператори, які вирішують основну проблему SQL-ін'єкцій: плутанину даних і коду. Наприклад, поле адреси електронної пошти слід розглядати як дані. Підготовлені оператори явно відокремлюють дані від коду, роблячи SQL-ін'єкції неможливими. З цією метою SQL-вирази містять заповнювачі, а не фактичні дані. Фрагменти даних, які вставляються замість заповнювачів, надсилаються окремо до бази даних. Вихідний код на рис. 4.7 ілюструє підготовлені твердження.

У рядках 2–4 створіть підготовлене твердження 'stmt', використовуючи знаки запитання ('?') як заповнювачі для даних. У рядку 5 знакам питання присвоюються фактичні значення (оголошуючи їх двома рядками). Після цього запит виконується по рядку 6. Дані будуть використовуватися системою баз даних в місцях, позначених заповнювачами.

```
1 <?php
2 $stmt = $db->prepare(
3     "SELECT id, last_active FROM users
4      WHERE email = ? AND password = ?")
5 $stmt->bind_param("ss", $email, $password);
6 $result = $stmt->execute();
7 ?>
```

Рис.4.7. PHP-код з підготовленою інструкцією для захисту від SQL-ін'єкційних атак

У реальних додатках SQL-ін'єкції з'являються, особливо коли SQL-вирази будуються динамічно, наприклад, коли умови додаються і видаляються на основі введених користувачем даних. Оскільки SQL-ін'єкцій легко уникнути, їх

виникнення є показником недостатньої освіти розробників з безпеки.

Особливої важливості для попередження вразливостей при розробленні та розповсюдженні веб-додатків набуває розкриття повної інформації про розроблений продукт для споживачів. Разом з тим, розкриття інформації не позбавлене недоліків. Деяке програмне забезпечення розповсюджується різними організаціями, які можуть випускати виправлення в різний час. Це стосується дистрибутивів Linux, які містять тисячі різних програмних пакетів. Виправлення, випущене одним дистрибутивом Linux, може надати інформацію про вразливість, яка потім може бути використана зловмисниками для атаки на користувачів інших дистрибутивів Linux, які ще не випустили виправлення. Крім того, чим більше людей беруть участь у розробці та розповсюдженні виправлення, тим більша ймовірність того, що інформація про вразливість просочиться до того, як виправлення буде відправлено.

Постачальники повинні дотримуватися встановлених найкращих практик для адекватної обробки вразливостей.

По-перше, вони повинні надати спеціальну контактну особу з безпеки на своєму веб-сайті, щоб гарантувати, що звіти про вразливості потраплять до потрібної групи в організації. В іншому випадку допоміжний персонал, який не навчений читати звіти про технічну безпеку, може проігнорувати ці звіти через непорозуміння. Крім того, рекомендується надати публічний ключ (пор. 2.4.3) для обміну зашифрованими повідомленнями з контактом безпеки, наприклад за допомогою OpenPGP.

По-друге, постачальник повинен підтвердити отримання звіту про вразливість і після розслідування проблеми підтвердити проблему (якщо вона дійсна).

По-третє, очікується, що постачальник запропонує графік скоординованого випуску виправлення та звіту.

Шукачі вразливостей інвестують свій час, щоб зробити користувачів програмного забезпечення постачальника більш безпечними. Правові погрози у відповідь на повідомлення вважаються аморальними і можуть призвести до

ефекту Стрейзанд, тобто спроба приховати або цензурувати деяку інформацію призводить до ненавмисного поширення інформації ширше. Сьогодні це часто відбувається через соціальні мережі та призводить до негативного розголосу постачальника програмного забезпечення.

Замість юридичних загроз спільнота безпеки закликає постачальників бути прозорими щодо проблем безпеки у своїх продуктах. Крім того, постачальники повинні надавати якомога більше інформації, щоб дозволити своїм користувачам точно оцінити будь-які ризики, на які вони можуть наражатися. Найкращою практикою вважається швидке надання виправлення. Крім того, деякі постачальники пропонують програму винагороди за виявлення помилок, яка надає повідомникам про вразливості грошову компенсацію.

Висновок до розділу 4. Зміст розділу відображає результати досліджень та аналізу, а також пропозиції для подальшого удосконалення та застосування комплексної методики виявлення, аналізу та оцінювання вразливостей веб-додатків. Визначені та обґрунтовані критерії оцінювання стратегій веб-безпеки.

Встановлено, що вивчення та оцінка ефективності стратегій забезпечення безпеки є важливим етапом для визначення працездатності та застосовності розроблених методик. У розділі проведено оцінку результатів застосування комплексної методики на реальних веб-додатках, визначено її переваги та недоліки.

За результатами дослідження запропоновані рекомендації постачальникам щодо оброблення вразливостей веб-додатків та обміну досвідом із застосуванням штучного інтелекту.

Надані конкретні пропозиції та рекомендації щодо практичного впровадження розробленої комплексної методики в реальному середовищі. Розглянуті аспекти інтеграції методики в процес розробки та підтримки веб-додатків, а також визначені оптимальні стратегії впровадження з урахуванням конкретних вимог та характеристик організації.

ВИСНОВКИ

У магістерській роботі висвітлений комплексний підхід до забезпечення безпеки веб-додатків, розглядаючи широкий спектр аспектів від основних понять безпеки та характеристик веб-додатків до вивчення типових вразливостей, сучасних методів виявлення та аналізу цих вразливостей.

Аналіз теоретичних аспектів безпеки веб-додатків, основних понять, характеристик та структури веб-додатків дозволив сконцентрувати увагу на проведенні аналізу технологій виявлення аналізу та оцінювання вразливостей сучасних веб-додатків.

Встановлено, що розвиток технологій впливає як на стійкість безпеки веб-додатків так і на загрози безпеці і потребує наукових досліджень та розробок щодо нових підходів, способів та методів застосування штучного інтелекту для створення ефективної веб-безпеки додатків.

У роботі вперше розроблено та представлено комплексну методику, яка враховує різноманітні аспекти безпеки на різних етапах життєвого циклу веб-додатка та поєднує вимоги міжнародних стандартів, вітчизняного законодавства та механізмів захисту веб-додатків на основі штучного інтелекту.

Розроблений комплексний підхід до безпеки веб-додатків є перспективним та ефективним. Використання штучного інтелекту у поєднанні з сучасними методами дозволяє не лише виявляти існуючі вразливості, але й прогнозувати потенційні ризики, адаптуючись до змін у сфері кібербезпеки.

Встановлено, що вивчення та оцінка ефективності стратегій забезпечення безпеки є важливим етапом для визначення працездатності та застосовності розроблених методик.

За результатами дослідження запропоновані рекомендації постачальникам щодо оброблення вразливостей веб-додатків та обміну досвідом із застосуванням штучного інтелекту.

Практичне впровадження розробленої методики може значно підвищити рівень безпеки веб-додатків, сприяючи уникненню атак та зменшенню ризиків для користувачів та організацій. Результати даної роботи можуть слугувати

підґрунтям для подальших досліджень та вдосконалення стратегій захисту веб-додатків у змінних умовах кіберзагроз.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бондаренко О., Ушкаленко І. Безпека Web-додатків: актуальні проблеми та їх аналіз. *Формування ринкової економіки в Україні*, 38, 2017. С. 28-36.

2. Трофименко О.Г., Козін О.Б. Веб-дизайн та HTML-програмування: навч.- метод. посібник. Одеса: Фенікс, 2017. 194 с.

3. Jazayeri M. Mesnage CS., Rose . Modern Web Application Development. Emerging Methods, Technologies, and Process Management in Software Engineering. *John Wiley and Sons Inc*., 2007. С. 131-147. URL: https://doi.org/10.1002/9780470238103.ch7.

4. Herrmann, D., Pridöhl, H. Basic Concepts and Models of Cybersecurity. In: Christen, M., Gordijn, B., Loi, M. (eds) The Ethics of Cybersecurity. *The International Library of Ethics, Law and Technology*, vol 21. 2020. Springer, Cham. URL: https://doi.org/10.1007/978-3-030-29053-5_2.

5. Waked L., Mannan M., Youssef A. *To intercept or not to intercept: Analyzing tls interception in network appliances.* Proceedings of the 2018 on Asia Conference on Computer and Communications Security. 2018. pp. 399-412. URL: https://doi.org/10.1145/3196494.3196528.

6. Башовий, В.М., Стаценко В.В., Стаценко Д.В. Визначення швидкості роботи сучасних фреймворків для створення web-інтерфейсів. *Технології та інжиніринг*. 2022. № 4 (9). С. 9 –16. URL: https://api.semanticscholar.org/CorpusID:258837238.

7. Golian, N.V. Golian V.V., Afanasieva I.V. Black and white-box unit testing for web applications. Вісник Національного технічного університету «ХПІ». Серія: *Системний аналіз, управління та інформаційні технології*. № 1 (7). 2022. С. 79 - 83. URL: https://doi.org/10.20998/2079-0023.2022.01.13.

8. Nunes P. J. Blended Security Analysis for Web Applications: Techniques and Tools: дис. … 00500: Universidade de Coimbra, 2022. 247 p.

9. Sotnik, S. Al-Sherrawi, Mohannad H. Saadoon, Ali Malik. Information model of plastic products formation process duration by injection molding. *International Journal of Mechanical Engineering and Technology*. 2018. Vol. 9(3). pp. 357–366. Available online at http://www.iaeme.com/IJMET/issues.asp?JType=IJMET&VType=9&IType=3.

10. Sotnik, S. Nevliudova V., Malaya I. Developing the information search system for selecting the moulds forming elements. *Innovative technologies and scientific solutions for industries (ITSSI).* 2(2). 2017. p. 86–92. URL: https://doi.org/10.30837/2522-9818.2017.2.086.

11. Mohammad, A. Informational and Structural-Parametric Models of Inductions Micromotors. *IOSR Journal of Electrical and Electronics Engineering .(IOSR-JEEE)*. 2018. p. 66-76. URL: https://doi.org/10.9790/1676-1302016676.

12. X. de Carné de Carnavalet, van P. Oorschot. A Survey and Analysis of TLS Interception Mechanisms and Motivations: Exploring how end-to-end TLS is made “end-to-me” for web traffic. *ACM Computing Surveys*. 55:13s. (1-40). Online publication date: 31-Dec-2024. URL: https://doi.org/10.1145/3580522.

13. Анипченко О. Є., Щавінський Ю.В. Дослідження ступеня захищеності Web-додатків на основі аналізу їх структури та інформаційного наповнення. Сучасний захист інформації №3(51), Київ, 2022. С. 39-47. URL: https://doi.org/10.31673/2409-7292.2022.033947.

14. Шило С.Г. Інформаційні системи та технології : навч. посіб. Х. : ХНЕУ, 2013. 219 с.

15. Кібербезпека : сучасні технології захисту. Навчальний посібник для студентів вищих навчальних закладів. Львів: «Новий Світ-2000», 2020 . 678 с.

16. Про інформацію: Закон України, поточна редакція від 16.07.2020 р. №2657-XII. Верховна Рада України. Офіц. вид. URL: https://zakon.rada.gov.ua/laws/show/2657-12#Text. (дата звернення: 15.11.2023).

17. Про захист інформації в інформаційно-телекомунікаційних системах: Закон України, поточна редакція від 04.07.2020 р. №80/94-ВР. Верховна Рада

України. Офіц. вид. URL: https://zakon.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80#Text. (дата звернення: 15.11.2023).

18. Захист інформації в комп'ютерних системах та мережах: навч. посіб. / С.Г.Семенов, А.О.Подорожняк, О.І.Баленко, С.Ю.Гавриленко. Х.: НТУ «ХПІ», 2014.– 251 с.

19. Про захист персональних даних: Закон України №2297-VI від 23.04.2021. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text. (дата звернення: 15.11.2023).

20. Про основні засади забезпечення кібербезпеки України: Закон України. (Відомості Верховної Ради (ВВР), 2017, № 45, ст.403). URL: https://zakon.rada.gov.ua/go/2163-19. (дата звернення: 15.11.2023).

21. Rich C., Himanshu D., Lackey Z.. Hacking Exposed Web 2.0: Web 2.0 Security Secrets and Solutions. *McGraw-Hill Education*. December 2007. 258 p.

22. Бурячок В. Л., Складанний П.М., Лукова-Чуйко Н.В. Інформаційний та кіберпростори: проблеми безпеки, методи та засоби боротьби: посібник. К. : ДУТ КНУ, 2016. 178 с.

23. Бурячок В.Л., Толубко В.Б., Хорошко В.О., Толюпа С.В. Інформаційна та кібербезпека: соціотехнічний аспект: підручник.– К.: ДУТ, 2015.– 288 с.

24. Якименко Ю.М., Савченко В.А., Легомінова С.В. Системний аналіз інформаційної безпеки: сучасні методи управління: підручник. Київ: ДУТ, 2022. -308 с.

25. Ashraf, S. Avoiding Vulnerabilities and Attacks with a Proactive Strategy for Web Applications. *Advances in Robotics and Mechanical Engineering*. Volume 3. Issue 2. 2021. pp. 263-271. URL: https://doi.org/10.32474/ARME.2021.03.000157.

26. Deineko Z. Confidentiality of Information when Using QR-Coding. *IJAISR*. Vol. 6 Issue 9, September. 2022. pp. 10-15. URL: https://openarchive.nure.ua/handle/document/20981.

27. Solanki J. Serverless Architecture – What It Is? Benefits, Limitations & Use cases Jignesh Solanki. 2023. URL: https://www.simform.com/blog/serverless-architecture-guide. (дата звернення: 06.11.2023).

28. Fowler M. Design - Who needs an architect? IEEE Software, vol. 20, no. 5. 2003. pp. 11-13. URL: https://ieeexplore.ieee.org/document/1231144. (дата звернення: 06.11.2023).

29. OWASP Top 10 Security Risks & Vulnerabilities URL: https://sucuri.net/guides/owasp-top-10-security-vulnerabilities-2020. (дата звернення: 06.10.2023).

30. Securing web applications: top OWASP threats and what to do about them URL: https://www.ptsecurity.com/ww-en/analytics/knowledge-base/securing-webapplications-top-owasp-threats-and-what-to-do-about-them/. (дата звернення: 06.10.2023).

31. OWASP Web Security Testing Guide. URL: https://owasp.org/www-project-web-security-testing-guide/. (дата звернення: 06.10.2023).

32. ISO/IEC 27001:2013. Information technology – Security techniques – Information security management systems – Requirements. URL: http://www.itgivernance.co.uk/standards.arx.

33. Топ-10 OWASP -2017. Десять самих критичних загроз безпеці веб-додатків. URL: https://owasp.org/www-pdfarchive/OWASP_Top_10-2017-ru.pdf. (дата звернення: 07.12.2023).

34. NIST Special Publication 800-63B. Digital Identity Guidelines. Authentication and Lifecycle Management. URL: https://pages.nist.gov/800-63-3/sp800-63b.html.

35. Замула О.А., Черниш В.І. Аналіз міжнародних стандартів в галузі оцінювання ризиків ін формаційної безпеки. *Системи обробки інформації: зб. наук. пр. Х.*: ХУПС, 2011. Вип. 2 (92). С.53-56. URL: https://www.hups.mil.gov.ua/hups/periodic-app/article/8219.

36. НД ТЗІ 2.5-010-03. Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу. URL: https://tzi.com.ua/downloads/2.5-010-03.pdf. (дата звернення: 24.11.2023).

37. НД ТЗІ 3.7-003-05. Порядок проведення робіт із створення комплексної системи захисту інформації в інформаційно-телекомунікаційній системі". URL:

http://www.dsszzi.gov.ua/dstszi/control/uk/publish/article?art_id=46074&cat_id=38835. (дата звернення: 24.11.2023).

38. Гавриленко О.В. Відповідність національної нормативної бази у сфері технічного захисту інформації міжнародним стандартам: зіставлення документів, шляхи гармонізації. Матеріали XVII Міжнародної науково-практичної конференції «Безпека інформації у інформаційно-телекомунікаційних системах», м.Київ, 2015. URL: https://ela.kpi.ua/bitstream/123456789/9848/1/16_p6.pdf.

39. Sen J. *A Robust Mechanism for Defending Distributed Denial OF Service Attacks on Web Servers*. International Journal of Network Security & Its Applications (IJNSA). 2011, March. Vol. 3, N 2. pp. 162-179. URL: https://doi.org/10.5121/ijnsa.2011.3213.

40. Bhavani A.B. Cross-site Scripting Attacks on Android WebView. *International Journal of Computer Science and Network*. 2013. Vol. 2, Issue 2. 5 p.: URL: http://ijcsn.org/IJCSN-2013/2-2/IJCSN-2013-2-2-03.pdf.

41. Cuff P. Distributed channel synthesis. IEEE. *Trans. Inf. Theory*. 2013. Vol. 59(11). P. 7071-7096. URL: https://doi.org/10.1109/TIT.2013.2279330.

42. Schieler C. Cuff P. Rate-distortion theory forsecrecy systems. *IEEE Trans. on Inf. Theory*. – 2014. – Vol. 66(12). – P.7584-7605. URL: https://doi.org/10.1109/TIT.2014.2365175.

43. Wasyihun Sema Admass, Yirga Yayeh Munaye, Abebe Abeshu Diro, Cyber security: State of the art, challenges and future directions. *Cyber Security and Applications*. Volume 2, 2024, 100031, URL: https://doi.org/10.1016/j.csa.2023.100031.

44. Ramanpreet Kaur, Dušan Gabrijelčič, Tomaž Klobučar. Artificial intelligence for cybersecurity: Literature review and future research directions. *Information Fusion*. Volume 97. 2023. 101804. URL: https://doi.org/10.1016/j.inffus.2023.101804.

45. Євтєєв Д. Методи обходу Web Application Firewall. URL: http://www.ptsecurity.ru/download/PTdevteev-CC-WAF.pdf. (дата звернення:

13.10.2023).

46. Hoffman A.. Web Application Security Exploitation and Countermeasures for Modern Web Applications. 2020. O'Reilly Media, Inc. 330 p.

47. Voitovych, O. P., O. S. Yuvkovetskyi, and L. M. Kupershtein. SQL injection prevention system. 2016 International Conference Radio Electronics & Info Communications (UkrMiCo). IEEE, 2016. URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/27968/45nigxju2tpe63tv5iy6aa8jor5pguaiic.pdf?sequence=1&isAllowed=y. (дата звернення: 16.10.2023).

48. Cheng Y.-L., Lee C.-Y.,. Huang Y.-L, Buckner C.A., Lafrenie R.M., J.A. Dénommée, Caswell J.M., Want D.A., Gan G.G., Leong Y.C., Bee P.C., E. Chin, A.K.H. Teh, S. Picco, L. Villegas, F. Tonelli, M. Merlo, J. Rigau, D. Diaz, …, R.H.J. Mathijssen. Smart health and cybersecurity in the era of artificial intelligence. *Intech.* 11 2016. p. 13. https://www.intechopen.com/books/advanced-biometric-technologies/liveness-detection-in-biometrics. (дата звернення: 16.10.2023).

49. Sahin C.S. General Framework for Evaluating Password Complexity and Strength / C.S. Sahin, R. Lychev, N. Wagner. 11 p. Режим доступу в Інтернет: http://arxiv.org/abs/1512.05814. (дата звернення:13.11.2023).

50. Website Security Statistics Report: 2022. WhiteHat Security, 2022. 31 p. URL: https://info.whitehatsec.com/Website-Stats-Report-2022.html. (дата звернення: 07.10.2023).

51. Zhang X, Cheng Q, Li Y. LaTLS: A Lattice-Based TLS Proxy Protocol. *Chinese Journal of Electronics.* 31:2. 2022. pp. 313-321. URL: https://doi.org/10.1049/cje.2018.00.357.

52. Humayun M., Niazi M., Jhanjhi N., Alshayeb M., Mahmood S.. Cyber security threats and vulnerabilities: a systematic mapping study. *Arab. J. Sci. Eng.,* 45 (4) .2020. pp. 3171-3189, URL: https://doi.org/10.1007/s13369-019-04319-2.

53. Website Security Statistics Report: 2017. WhiteHat Security, 2017. 30 p. URL: https://info.whitehatsec.com/Website-Stats-Report-2017.html . (дата звернення: 22.11.2023).

54. Best language for web application development. steelwiki. URL:

https://steelkiwi.com/blog/best-languages-for-web-applicationdevelopment. (дата звернення: 12.11.2023).

55. Cybersecurity threatscape: Q3 2019. URL: https://www.ptsecurity.com/ww-en/analytics/cybersecurity-threatscape-2019-q3/ (дата звернення: 09.11.2023).

56. Gunasundaram Rajesh. ASP.NET WEB API Security Essentials Packt Publishing, 2015. 152 p.

57. Lakshmiraghavan B. Pro ASP.NET WEB API Security: Securing ASP.NET WEB API Apress, 2013. – 403 p.

58. The Web Application Security Consortium. URL: https://www.academia.Edu/11623665/Web_Application_Security_Consortium_Threat_Classification_WASC-TC_and_ISECOM_Open_Source_ Security_ Testing_ Methodology_Manual_OSSTMM_ (дата звернення: 07.11.2023).

59. Cummins C. et al. *Compiler fuzzing through deep learning*. Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis. 2018. pp. 95-105. URL: https://doi.org/10.1145/3213846.3213848.

60. Xu H. et al. *Dsmith: Compiler fuzzing through generative deep learning model with attention*. 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020. pp. 1-9. URL: https://doi.org/10.1186/s12964-019-0473-9.

61. Yazdinejad A., Kazemi M., Parizi R.M., Dehghantanha A., Karimipour H.. An ensemble deep learning model for cyber threat hunting in industrial internet of things. *Digit. Commun. Netw*, 9 (1). 2022. pp. 101-110. URL: https://doi.org/10.1016/j.dcan.2022.09.008.

62. Zhang Z., Ning H., Shi F., Farha F., Xu Y., Xu J., Zhang F., Choo K.K.R. Artificial intelligence in cyber security: research advances, challenges, and opportunities. *Artif. Intell. Rev*., 55 (2). 2022. pp. 1029-1053, URL: https://doi.org/10.1007/s10462-021-09976-0.

63. Liu S. et al. DeepBalance: Deep-learning and fuzzy oversampling for vulnerability detection. *IEEE Transactions on Fuzzy Systems.* 2019. Vol.. 28. №. 7. pp. 1329-1343. URL: https://doi.org/10.1109/DDCLS.2019.8908835.