Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра мультимедійних систем

Магістерська робота

Освітній ступінь – магістр

На тему: «ПРОАКТИВНИЙ ЗАХИСТ ВЕБ ЗАСТОСУНКІВ ВІД КІБЕРАТАК НА ФРОНТЕНДІ»

Виконав студент 2-го року навчання,
Спеціальності
121 Інженерія програмного забезпечення

Винник Андрій Ігорович

Керівник Олецький О.В.
Кандидат технічних наук, доцент

Рецензент _____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою _____

Секретар ЕК _____

«___» _____ 20___р.

Київ – 2024

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра мультимедійних систем факультету інформатики

ЗАТВЕРДЖУЮ

Кандидат технічних наук, доцент

Олецький О.В.

_____

(підпис)

“___” _____ 2024 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на магістерську роботу

студенту Виннику А.І. факультету інформатики 2 курсу

ТЕМА: Проактивний захист веб застосунків від кібератак на фронтенді

Зміст ТЧ до курсової роботи:

1. Індивідуальне завдання
2. Вступ
3. Огляд сфери розробки веб-застосунків
4. Опис загроз для кібератак на фронтенді
5. Опис проактивного захисту від кібератак
6. Імплементація проактивного захисту
7. Висновки
8. Список літератури
9. Додатки

Дата видачі “___” _____ 2023 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: Проактивний захист веб застосунків від кібератак на фронтенді

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на дипломну роботу.	18.10.2023	
2.	Огляд технічної літератури за темою роботи.	15.11.2023	
3.	Ознайомлення із загрозами на фронтенді	25.01.2024	
4.	Огляд проактивних методів захисту	17.02.2024	
5.	Обговорення предметної області з керівником	22.02.2024	
6.	Імплементація проактивного захисту у веб застосунку	06.04.2024	
7.	Написання текстової частини	15.05.2024	
8.	Захист курсової роботи	03.06.2024	

Зміст

УМОВНІ ПОЗНАЧЕННЯ 5

ВСТУП 6

1. ВАЖЛИВІСТЬ ПРОБЛЕМИ ЗАХИСТУ ВІД КІБЕРАТАК. 8

1.1. Загрози кібербезпеки на фронтенді 12
1.2. Механізми та наслідки кібератак на фронтенді 13
1.3. Збільшення кількості та складності атак 22

2. ПРОАКТИВНИЙ ЗАХИСТ ВІД КІБЕРАТАК НА ФРОНТЕНДІ 25

2.1. Переваги проактивного захисту 26
2.2. Методи проактивного захисту 27
2.3. Навчання та обізнаність 30

3. РОЗРОБКА ЗАСТОСУНКУ З ПРОАКТИВНИМ ЗАХИСТОМ НА ФРОНТЕНДІ 33

3.1. Вибір технологій та інструментів 33
3.1.1. Firebase 34
3.1.2. Angular 35
3.2. Архітектура застосунку 36
3.3. Проактивні методи захисту 39
3.3.1. Аутентифікація та авторизація 39
3.3.2. Захист від XSS атак 41
3.3.2.2. Content Security Policy 43
3.3.3. Вбудований захист Angular 44
3.3.3.1. Google ReCaptcha 47

ВИСНОВОК 49

ДОДАТКИ 51

ДЖЕРЕЛА 52

УМОВНІ ПОЗНАЧЕННЯ

ІКТ – Інформаційно -комунікаційні технології

XSS – Cross-Site Scripting

CSRF – Cross-Site Request Forgery

PCI DSS - Payment Card Industry Data Security Standard

GDPR - General Data Protection Regulation

JWT – Json Web Tokens

MITM – Man-in-the-middle

ВСТУП

У сучасному світі, де технології стрімко розвиваються, веб-застосунки стали невід'ємною частиною повсякденного життя, бізнесу та комунікацій. Проте з цим розвитком зростає і кількість кібератак, які загрожують безпеці та конфіденційності користувачів. Згідно з дослідженням компанії Cybersecurity Ventures, до 2025 року глобальні втрати від кіберзлочинності можуть досягти 10,5 трильйонів доларів щорічно. Ця статистика підкреслює актуальність дослідження методів проактивного захисту веб-застосунків, особливо на фронтенді, який часто стає першою мішенню для атак.

Вибір теми "Проактивний захист веб-застосунків від кібератак на фронтенді" обумовлений необхідністю розробки ефективних механізмів захисту, які можуть запобігти можливим загрозам ще до їх виникнення. Проактивні методи безпеки дозволяють не тільки реагувати на атаки, але й передбачати та запобігати їм, що значно підвищує загальний рівень безпеки веб-застосунків.

Ціллю даної роботи є дослідження веб-застосунків, котрі використовують фронтенд для взаємодії з користувачами. Це обумовлено великим попитом на веб сторінки та їх розповсюдженістю в ІКТ.

Предметом дослідження є проактивні методи та технології захисту фронтенд-частини веб-застосунків від кібератак. Це включає аналіз різних видів атак, таких як XSS (міжсайтовий скриптинг), CSRF (міжсайтові підробки запитів), SQL-ін'єкції та інших, а також розробку та впровадження заходів, спрямованих на запобігання цим атакам.

Метою даного дослідження є розробка та впровадження проактивних методів захисту фронтенд-частини веб-застосунків від кібератак. Для досягнення цієї мети необхідно вирішити наступні завдання:

1. Провести аналіз існуючих методів захисту веб-застосунків та їх ефективності.
2. Ідентифікувати основні загрози та вразливості фронтенд-частини веб-застосунків.

3. Розробити проактивні методи та технології захисту, які зможуть ефективно запобігати і мінімізувати ризики кібератак.
4. Провести експериментальне тестування розроблених методів на реальних веб-застосунках.
5. Оцінити ефективність запропонованих рішень та надати рекомендації щодо їх впровадження у практику.

Основними джерелами для даного дослідження є наукові статті, дослідження та аналітичні звіти, присвячені проблемам кібербезпеки та захисту веб-застосунків. Зокрема, значну увагу було приділено дослідженням OWASP (Open Web Application Security Project), які регулярно публікують звіти про найпоширеніші вразливості веб-застосунків.

Методологічною основою даного дослідження є комплексний підхід, який включає аналіз літератури, експериментальні дослідження, моделювання та тестування розроблених методів захисту.

Таким чином, дослідження спрямоване на розробку ефективних проактивних заходів захисту веб-застосунків, що дозволить знизити ризики кібератак та підвищити загальний рівень безпеки інформаційних систем**.**

1. Важливість проблеми захисту від кібератак.

В сучасному цифровому віці інформаційні технології стали невід'ємною частиною нашого повсякденного життя. Проте разом з розвитком інтернету та технологій з'явилася й небезпека кібератак. Кіберзагрози, такі як хакерські атаки, віруси, фішинг та інші види кіберзлочинності, стали серйозним викликом для інформаційної безпеки. У зв'язку з цим стає важливим розуміння важливості захисту від кібератак та прийняття ефективних заходів для запобігання їхнім наслідкам.

Протягом останніх кількох років експерти та політики виражають зростаючу занепокоєність щодо захисту від кібератак - намагань несанкціонованих осіб отримати доступ до ІКТ-систем, зазвичай з метою крадіжки, руйнування, шкоди або інших незаконних дій. Багато експертів очікують зростання кількості та серйозності кібератак у наступні кілька років. Дії щодо захисту ІКТ-систем та їх вмісту отримали назву кібербезпека, це широке і, можна сказати, дещо неоднозначне поняття, яке може бути корисним терміном, але схильне до точного визначення.

Кібербезпека це:

- набір дій та інших заходів, спрямованих на захист - від атак, руйнування чи інших загроз - комп'ютерів, комп'ютерних мереж, пов'язаного обладнання та пристроїв програмного забезпечення та інформації, яку вони містять і передають, включаючи програмне забезпечення та дані, а також інші складові кіберпростору;
- стан або якість захищеності від таких загроз;
- широке поле діяльності, спрямоване на впровадження та покращення цих дій і якості.

Більшість атак у кіберпросторі не мають значного впливу, але якщо атака виявиться успішною та спрямованою на ключові складові критичної інфраструктури, яка переважно належить приватному сектору, це може серйозно підірвати національну безпеку, економіку та безпеку громадян. Таким чином,

рідкісна, але успішна атака з великим впливом може мати більшу загрозу, ніж звичайна атака з низьким впливом.

Хоча кібератаки загалом вважаються дорогими для осіб та організацій, економічні наслідки їх дій складно оцінити, і оцінки витрат значно відрізняються. Часто цитується цифра щорічних витрат на глобальну економіку від кіберзлочинності - 400 мільярдів доларів, проте деякі експерти стверджують, що витрати зростають, особливо з розширенням ІКТ-інфраструктури через Інтернет речей та інших нових платформ. Витрати на кібершпигунство можуть бути ще складніше оцінити, але вони також вважаються значними.

Зростаюча складність та інтенсивність кібератак підкреслює важливість впровадження проактивної стратегії захисту систем інформаційних технологій. Важливо мати можливість реагувати на ситуацію та приймати необхідні заходи перед тим, як кібератака завдасть шкоди. Однак, жорсткі часові рамки ускладнюють використання методів експертної оцінки, що призводить до зростання значення автоматизованих процесів безпеки.

Засоби автоматизації повинні мати високий рівень інтелектуалізації, щоб вони можливо найкращим чином відповідали потребам операторів кібербезпеки. Одним з ключових аспектів їх розробки є створення процесної моделі об'єкту дослідження, на основі якої створюється відповідне програмне забезпечення. Проте, недостатнє розуміння терміну "інформація" може ускладнити аналіз процесів кіберзахисту та визначення їх характеристик.

Управління ризиками від кібератак включає усунення джерела загрози (наприклад, припинення роботи ботнетів або зменшення стимулів для кіберзлочинців); вирішення вразливостей шляхом зміцнення ІКТ-активів (наприклад, патчінг програмного забезпечення та підготовка персоналу); та зменшення впливу шляхом пом'якшення шкоди та відновлення функцій (наприклад, наявність резервних ресурсів для забезпечення неперервності операцій у відповідь на атаку). Оптимальний рівень зменшення ризику буде відрізнятися для різних секторів та організацій. Наприклад, рівень очікуваної

кібербезпеки може бути нижчим для компанії в розважальному секторі порівняно з банком, лікарнею або урядовим агентством.

Законодавчі та виконавчі ініціативи, зазвичай, переважно націлені на вирішення кількох важливих потреб у кібербезпеці на короткостроковій перспективі: запобігання кібер-катастрофам та кібершпигунству, зменшення наслідків успішних атак, покращення співпраці між секторами та всередині них, уточнення ролей та відповідальності федеральних агентств і боротьба з кіберзлочинністю. Проте ці потреби виникають у контексті складних викликів на довгостроковій перспективі, які пов'язані з дизайном, стимулами, згодою та середовищем. Фахівці часто наголошують на тому, що ефективна безпека має бути вбудована в дизайн ІКТ. Однак розробники традиційно приділяли більше уваги функціональності, а не безпеці, з економічних міркувань. Крім того, багато майбутніх потреб у безпеці неможливо передбачити, що створює складні завдання для дизайнерів. Структура економічних стимулів у сфері кібербезпеки називається викривленою або навіть поганою. Кіберзлочинність вважається дешевою, прибутковою та відносно безпечною для злочинців. У порівнянні з цим, кібербезпека може бути дорогим та недосконалим за своєю природою, а економічні вигоди від інвестицій часто неочевидні. Кібербезпека має різне значення для різних зацікавлених сторін, і часто не існує загальної згоди щодо її значення, реалізації та ризиків. Існують значні культурні перешкоди перед досягненням згоди, як не тільки між секторами, так і всередині секторів та навіть організацій. Кіберпростір називають найшвидше розвиваючимся технологічним простором в історії людства, і він постійно змінюється як за масштабом, так і за характером. Нові технології, такі як соціальні медіа, мобільні обчислення, великі дані, хмарні обчислення та Інтернет речей, додатково ускладнюють загрозливе середовище, але також можуть створювати можливості для покращення кібербезпеки.

Фронтенд веб-додатків - це та частина веб-додатка, яка відповідає за взаємодію з користувачем та відображення контенту на екрані. Вона включає в себе всі компоненти, які відображаються у веб-браузері користувача і з якими

користувач може взаємодіяти, такі як кнопки, текстові поля, меню, списки, форми та інші елементи інтерфейсу.

Основні складові фронтенду веб-додатків.

Складова	**Опис**
HTML (Hypertext Markup Language)	Використовується для створення структури та відображення контенту веб-сторінок. Визначає, які елементи будуть на сторінці, їхню структуру та розташування.
CSS (Cascading Style Sheets)	Відповідає за оформлення та стиль веб-сторінок. Дозволяє задавати кольори, шрифти, розміри, відступи та інші властивості елементів сторінки, щоб зробити її привабливішою та зручнішою для користувача.
JavaScript	Мова програмування, яка використовується для створення динамічного та інтерактивного веб-змісту. Дозволяє додавати взаємодію та функціональність до веб-сторінок, таку як анімація, обробка подій, валідація форм, взаємодія з сервером тощо.
Фреймворки та бібліотеки	React, Angular, Vue.js тощо. Дозволяють розробникам швидше та ефективніше створювати складні фронтенд-додатки шляхом надання готових компонентів та інструментів для роботи зі станом додатка, маршрутизацією, керуванням станом тощо.

Таблиця 1.1

Фронтенд грає ключову роль у створенні зручного та привабливого користувацького досвіду. Якість його впливає на те, наскільки легко користувачі можуть взаємодіяти з веб-додатком та як вони сприймають його. Ефективний фронтенд дозволяє створювати швидкі та реагуючі веб-додатки, що є важливим

для забезпечення задоволення користувачів та зниження відскоку. Фронтенд також грає роль у забезпеченні безпеки веб-додатків. Врахування безпекових аспектів при розробці фронтенду може допомогти уникнути загроз кібербезпеки, таких як XSS або CSRF.

1.1. Загрози кібербезпеки на фронтенді

Фронтенд веб-додатків, що відповідає за відображення та взаємодію з користувачем, стає об'єктом зростаючого інтересу для зловмисників у кіберпросторі. Зловмисники постійно шукають нові способи атак на клієнтську частину веб-додатків, щоб отримати доступ до конфіденційної інформації користувачів або вплинути на їх дії.

Клієнстька частина веб-додатків є одним з головних цілей для зловмисників, оскільки це інтерфейс, з яким користувачі взаємодіють. Атаки на фронтенд можуть призвести до крадіжки конфіденційної інформації, зміни вмісту веб-сторінок, або навіть до контролю над веб-додатком.[12]

Основні типи кіберзагроз на фронтенді.[12]

Тип загрози	**Опис**
Cross-Site Scripting (XSS)	Зловмисники вводять шкідливий JavaScript-код до веб-сторінки, щоб отримати доступ до конфіденційних даних користувача або керувати його браузером.
Cross-Site Request Forgery (CSRF)	Зловмисники змушують авторизованого користувача виконати небажані дії, наприклад, переказ коштів або зміну пароля.
Content Injection	Зловмисники вводять шкідливий контент до веб-сторінки, щоб змінити її вміст або перенаправити користувача на шкідливий веб-сайт.

Man-in-the-Middle Attacks	Зловмисники перехоплюють трафік між користувачем і веб-сервером, щоб перехопити дані або ввести шкідливий код.
SQL Injection	Зловмисники вводять шкідливий SQL-код до веб-додатку, щоб отримати доступ до даних бази даних або змінити їх.

Таблиця 1.2

Компрометація фронтенду може призвести до пошкодження репутації вашої компанії або бренду. Наприклад, якщо зловмисники змінять вміст вашого веб-сайту на неприпустимий або образливий, це може негативно позначитися на сприйнятті вашоєї компанії користувачами. Багато веб-додатків зберігають конфіденційні дані користувачів, такі як паролі, особисті дані та фінансові інформації. Компрометація фронтенду призводить до витоку цих даних, що може мати серйозні наслідки для користувачів та компаній.

Саме тому варто слідкувати і покращувати кібербезпеку фронтенду. Кібербезпека фронтенду - це галузь інформаційної безпеки, яка спеціалізується на захисті клієнтської частини веб-додатків та веб-сайтів від різних кіберзагроз. Вона охоплює широкий спектр заходів та технологій, спрямованих на запобігання вразливостям, використання захисних механізмів та реагування на потенційні атаки. Посилення заходів безпеки на рівні фронтенду допомагає зменшити ризики та забезпечити надійний захист від потенційних кіберзагроз.

1.2. Механізми та наслідки кібератак на фронтенді

Cross-Site Scripting (XSS) – це атака, під час якої зловмисник впроваджує шкідливі виконувані сценарії в код довіреної програми або веб-сайту. [2]

Зловмисники часто ініціюють атаку XSS, надсилаючи зловмисне посилання користувачеві та спонукаючи користувача натиснути його. Якщо додаток або веб-сайт не очищають належним чином дані, зловмисне посилання виконує вибраний

зловмисником код у системі користувача. У результаті зловмисник може викрасти файл cookie активного сеансу користувача.

XSS атаки відбуваються у випадку, коли дані, які надходять до веб-програми через ненадійне джерело, зазвичай через веб-запит, включаються у динамічний контент, що надсилається користувачеві без попередньої перевірки на наявність шкідливого вмісту. Цей шкідливий вміст, який передається веб-переглядачу, часто представлений у вигляді сегмента JavaScript, але також може містити HTML, Flash або будь-який інший тип коду, що здатний виконуватися браузером. Широкий спектр XSS атак майже необмежений, але зазвичай вони спрямовані на зловживання особистою інформацією користувачів, такою як файли cookie або інші дані сеансу, перенаправлення жертви на веб-контент, що контролюється зловмисником, або виконання інших шкідливих дій на комп'ютері користувача під виглядом вразливого веб-сайту. [2]

Ось приклад, як саме працює даний вид атаки:

<script>i=new/**/Image();isrc=http://evilwebsite.com/log.php?'+document.cookie+' '+document.location</script>

Хоча зазвичай зловмисники використовують JavaScript, XSS може працювати за допомогою будь-якої мови клієнта, що було сказано раніше. Щоб здійснити атаку міжсайтового сценарію, зловмисник вставляє шкідливий скрипт у введені користувачем дані. Зловмисники також можуть здійснити атаку, змінивши запит, тобто якщо веб-програма вразлива до атак XSS, введені користувачем дані виконуються як код, наприклад, у наведеному нижче запиті сценарій відображає вікно повідомлення з текстом «xss». [2-3]

http://www.site.com/page.php?var=<script>alert('xss');</script>

Існує багато способів ініціювати атаку XSS. Наприклад, виконання може запускатися автоматично, коли сторінка завантажується або коли користувач наводить курсор на певні елементи сторінки (наприклад, гіперпосилання). [2-3]

Потенційні наслідки атак міжсайтових сценаріїв включають:

- Захоплення натискань клавіш користувача
- Переспрямування користувача на шкідливий веб-сайт

– Запуск експлойтів на основі веб-браузера (наприклад, збій браузера)

– Отримання інформації про файли cookie користувача, який увійшов на веб-сайт, таким чином скомпрометувавши обліковий запис жертви

У деяких випадках XSS-атака призводить до повної компрометації облікового запису жертви. Зловмисники можуть обманом змусити користувачів ввести облікові дані у підробленій формі, яка надає всю інформацію зловмиснику.[2-3]

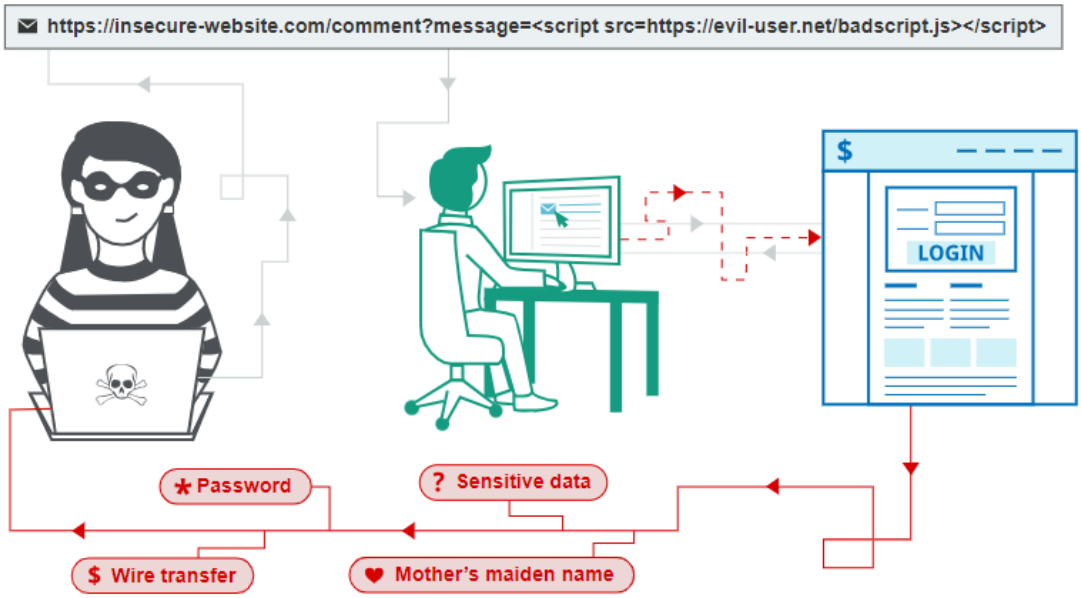

Рис. 1.3 Як працює XSS?

Cross-Site Request Forgery (CSRF) – це тип атаки, що змушує автентифікованих користувачів виконувати непотрібні запити до веб-додатку, на якому вони вже автентифіковані. Ці атаки використовують довіру веб-додатку до автентифікованих користувачів. На відміну від атак міжсайтового сценарію (XSS), де використовується довіра користувача до конкретного веб-додатку, атака CSRF використовує вразливість у веб-додатку, яка не дозволяє відрізнити запити, створені автентифікованим користувачем від тих, що створені без його згоди. [4]

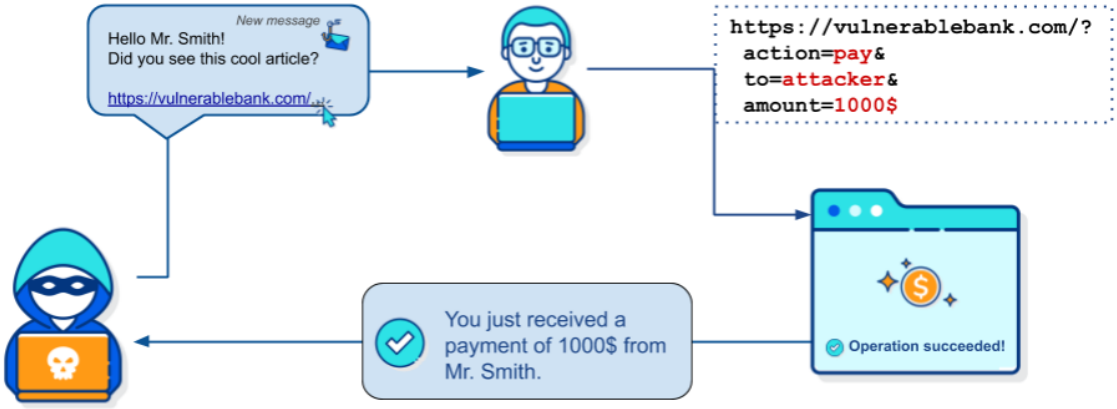

Рис. 1.4 Як працює CSRF.

Мета атак CSRF це змусити користувача виконати небажані дії, надсилаючи запит на цільовий веб-сайт, прикладом таких дій може бути переказ коштів з банківського рахунку, зміна пароля, опублікування повідомлення, видалення файлу, надання доступу до конфіденційних даних.

Зловмисники часто використовують платформи соціальної інженерії для запуску атак CSRF. Цей вид атаки включає обман жертви за допомогою спеціально створеної URL-адреси, яка містить незаконний запит до певної веб-програми. Після того, як жертва натискає на цю URL-адресу, її браузер надсилає зловмисно створений запит до цільової веб-програми. Цей запит також може включати облікові дані, такі як файли cookie сеансу користувача, пов’язані зі специфічним веб-сайтом. Якщо користувач має активний сеанс з цільовою веб-програмою, то програма розглядає цей новий запит як авторизований запит, поданий самим користувачем. Таким чином, зловмиснику вдається ефективно використати вразливість CSRF веб-додатку. [4-5]

Атаки CSRF спрямовані на веб-програми, які не можуть відрізнити справжні запити від підроблених, керованих зловмисником. Існує безліч способів, за допомогою яких зловмисники можуть намагатися використати цю вразливість.

Щоб успішно запустити такий вид атаки необхідно:

1. Зловмисник повинен створити URL-адресу експлойту.
2. Зловмисник також повинен обманом змусити користувача натиснути URL-адресу експлойту.

3. Користувачу потрібен активний сеанс із сайт.com.

Атаки CSRF націлюються на функціональність, яка викликає зміну стану на сервері, таку як зміна електронної адреси або пароля жертви, або покупка чогось.Зловмисник може використовувати CSRF для отримання особистих даних жертви за допомогою спеціальної форми атаки, відомої як атака CSRF на вхід. Зловмисник змушує невідентифікованого користувача увійти до облікового запису, яким він керує. Якщо жертва цього не помічає, вона може додати особисті дані, такі як інформація про кредитну карту, до облікового запису. Потім зловмисник може увійти до облікового запису, щоб переглянути ці дані, а також історію дій жертви в веб-застосунку.[4]

Іноді можливо зберегти атаку CSRF на самому вразливому сайті. Такі вразливості називаються "збереженими вадами CSRF". Це можна зробити просто, зберігши тег IMG або IFRAME в полі, яке приймає HTML, або складнішими атаками перехоплення міжсайтового скриптінгу. Якщо атака може зберегти атаку CSRF на сайті, тоді важкість атаки зростає. Зокрема, ймовірність збільшується, оскільки жертва з більшою ймовірністю перегляне сторінку, що містить атаку, ніж довільну сторінку в Інтернеті. Ймовірність також збільшується, оскільки жертва впевнена в тому, що вже автентифікована на сайті.[4]

SQL injection полягає у вставці або "ін'єкції" SQL-запиту через вхідні дані від клієнта до програми. Успішна експлуатація SQL-ін'єкції може прочитати чутливі дані з бази даних, модифікувати дані бази даних (вставка/оновлення/видалення), виконувати адміністративні операції з базою даних (такі як вимкнення СКБД), відновлювати вміст певного файлу, що присутній на файловій системі СКБД, і у деяких випадках видача команд операційній системі. Атаки SQL-ін'єкції є типом ін'єкційних атак, у яких SQL-команди вставляються во вхідні дані площини даних з метою впливу на виконання заздалегідь визначених SQL-команд. [8]

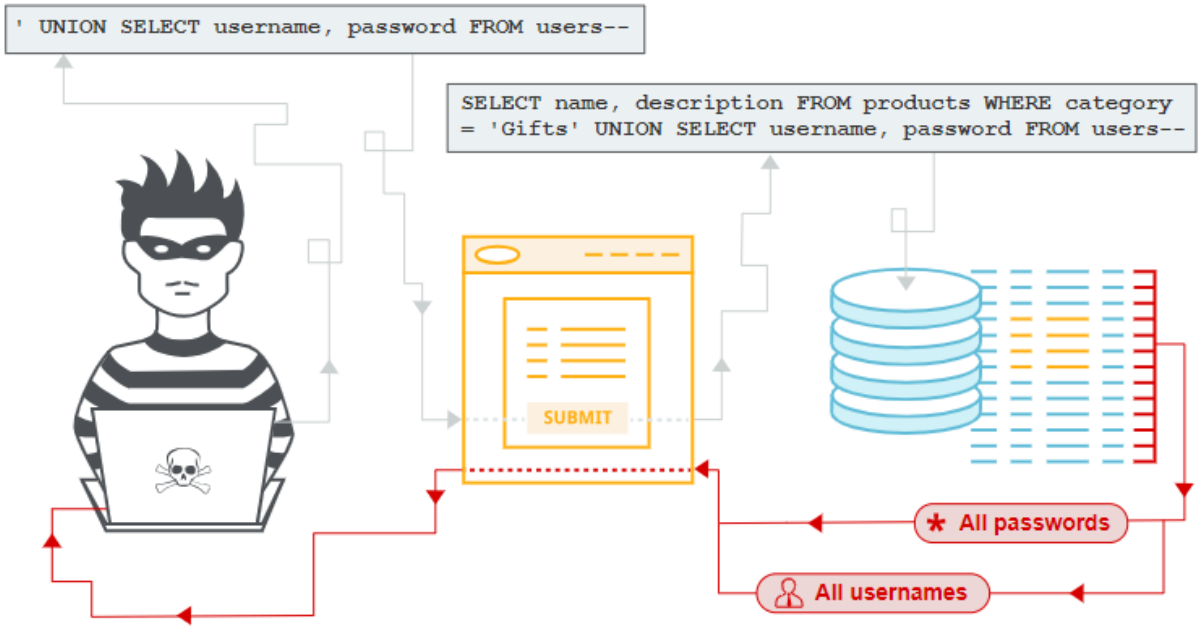

Рис. 1.5 Як працює SQL injection?[7]

SQL injection може мати серйозні наслідки для веб-додатку та його користувачів. До прикладу, хакери отримують доступ до даних користувачів що є конфіденційними, таких як адреси, імена, номери кредитних карток, паролі тощо, окрім цього вони матимуть можливість змінювати або видаляти дані, що зберігаються у веб-додатку, перевантажити веб-сервер або вивести його з ладу, що призведе до недоступності веб-додатку для користувачів. Кібератаки такого типу також можуть призвести до втрати довіри користувачів до веб-додатку та його власників.

Security risks in external scripts (зовнішні скрипти) - це фрагменти коду, які завантажуються з сторонніх веб-сайтів та виконуються у браузері користувача, вони часто використовуються для додавання функціональності до веб-сторінок, таких як карти, аналітика або рекламні банери.

Скрипти веб-сайтів відіграють ключову роль у забезпеченні їхньої динамічності та інтерактивності, що визначає зручність використання та функціональні можливості для користувачів. Без них сайт перетворюється на статичний об'єкт, що втрачає привабливість та корисність. Нажаль, разом із своєю корисністю, веб-скрипти є одними з найбільш вразливих елементів веб-сайтів. Швидкість написання скриптів може призвести до великих проблем з

безпекою. Для прискорення розробки часто використовується фреймворк jQuery. За даними досліджень, він застосовується на 74% веб-сайтів. Використання цього фреймворку може значно зекономити час, проте при недбалому підході може призвести до серйозних проблем з безпекою. Як і в інших фреймворках, вбудовані модулі та бібліотеки можуть містити вразливості.

Сучасні веб-сайти та програмне забезпечення створюють власні платформи для динамічного контенту, що генерується користувачами. Більшість мов програмування для вебу мають серверний характер, тому код виконується на сервері, що відкриває нові вектори атак для зловмисників. Не всі веб-скрипти розробляються досвідченими програмістами. Багато інтернет-проектів використовують безкоштовні або саморобні рішення, які не проходять ретельного тестування та не відповідають стандартам безпеки.

Зловмисники можуть використовувати зовнішні скрипти для вчинення кібератак на фронтенд веб-додатків. Наприклад, вводити шкідливий JavaScript-код до зовнішнього сценарію, який потім виконується у браузері користувача. Це може дати зловмисникам доступ до конфіденційних даних користувача, таких як паролі або номери кредитних карток. Перехоплювати та маніпулювати трафіком між веб-сервером та браузером користувача, це також можливо. Все це дозволяє зловмисникам вводити шкідливий код до зовнішніх скриптів, які потім виконуються у браузері користувача. Крім цього, зловмисники можуть вводити шкідливий контент, наприклад, текст, зображення або JavaScript-код, до зовнішнього сценарію, що в подальшому може змінити вміст веб-сторінки або перенаправити користувача на шкідливий веб-сайт.

Кібератаки, пов'язані із зовнішніми скриптами, можуть мати серйозні наслідки для користувачів та власників веб-додатків. Наприклад, такі атки дозволяють отримати доступ до конфіденційної інформації користувачів, такої як особисті дані, адреси, номери кредитних карток, паролі тощо, використовувати вкрадені дані кредитних карток для здійснення шахрайських операцій, що може призвести до фінансових втрат для користувачів та власників.

Кібератаки можуть викликати втрату довіри користувачів до веб-додатку та його власників через розголос про вразливості та інциденти безпеки, а також зловмисники можуть перевантажити веб-сервер або вивести його з ладу за допомогою кібератак, що може призвести до недоступності веб-додатку для користувачів і відмови у обслуговуванні.

Щоб уникнути проблем з безпекою, рекомендується звертатися до досвідчених розробників для створення веб-скриптів, використовувати перевірені фреймворки та бібліотеки, ретельно тестувати код та регулярно оновлювати програмне забезпечення. Також важливо використовувати надійні хостинги, які пропонують додаткові рівні безпеки.

Broken access control (порушений контроль доступу) — це тип уразливості програми, яка дозволяє користувачам отримувати доступ до даних і функцій, до яких вони не повинні мати доступу. У більшості випадків атак Broken Access Control зловмисник користується слабким або нереалізованим контролем доступу в цільовій програмі.[15]

Існує багато форм уразливості Broken Access Control. Одним із прикладів є звичайний користувач веб-програми, який може отримати доступ до сторінки адміністратора через погану реалізацію контролю доступу. Тепер цей користувач може продовжувати виконувати такі завдання, як видалення всіх даних або доступ до конфіденційних даних користувачів, як-от адреси електронної пошти всіх користувачів.[11]

Сценарій №1: Додаток використовує неперевірені дані у виклику SQL для доступу до інформації про обліковий запис:

```
pstmt.setString(1, request.getParameter("acct"));
 ResultSet results = pstmt.executeQuery( );
```

Зловмисник лише змінює параметр 'acct' браузера, щоб надіслати будь-який номер облікового запису, який вони хочуть. Якщо дані не перевіряються коректно, зловмисник може отримати доступ до будь-якого облікового запису користувача.

```
https://example.com/app/accountInfo?acct=notmyacct
```

Сценарій №2: Зловмисник просто змушує браузери переходити на цільові URL-адреси. Для доступу до сторінки адміністратора потрібні права адміністратора.

https://example.com/app/getappInfo https://example.com/app/admin_getappInfo

Якщо непідтверджений користувач може отримати доступ до будь-якої сторінки, це вразливість. Якщо не адміністратор може отримати доступ до сторінки адміністратора, це також вразливість.

Розміщення конфіденційних даних у відкритому доступі є серйозною загрозою для безпеки веб-середовища, тобто багато веб-додатків та API не забезпечують належним рівнем захисту конфіденційних даних, таких як фінансові дані, медична інформація та особиста ідентифікація, що відкриває двері для зловмисників, які можуть використовувати ці дані для вчинення шахрайства з кредитними картками, крадіжки особистих даних та інших злочинів.

Одним із способів захисту конфіденційних даних є шифрування, яке забезпечує безпеку як у спокої, так і під час передачі. Проте, навіть із застосуванням шифрування, важливо вживати додаткових заходів обережності при обміні чутливою інформацією через браузер.

Крім того, інструменти зовнішнього вигляду, такі як препроцесори CSS, можуть становити загрозу для безпеки веб-додатків. Навіть якщо ці інструменти спрощують процес програмування шляхом надання можливості використання циклів, функцій та інших конструкцій, їх використання може призвести до виникнення шкідливого коду, оскільки браузери не розуміють синтаксису цих інструментів і інтерпретують їх у звичайний CSS.

Ще однією проблемою є уразливості, які можуть з'являтися в різних версіях програмного забезпечення. Ці "дірки" можуть бути виявлені в окремих виданнях веб-додатків, що створює загрозу для безпеки. Навіть якщо програмне забезпечення написане якісно, популярність додатку збільшує ймовірність виявлення нових вразливостей. Тому важливо своєчасно встановлювати всі оновлення для захисту від потенційних загроз.

Узагальнюючи, безпека конфіденційних даних у веб-середовищі вимагає поєднання різних заходів захисту, включаючи шифрування, обережність при обміні даними через браузер та уважне вивчення інструментів зовнішнього вигляду. Тільки комплексний підхід до кіберзахисту може гарантувати захист від потенційних загроз та забезпечити безпеку конфіденційних даних у веб-середовищі.

1.3. Збільшення кількості та складності атак

У сучасному цифровому світі веб-додатки стали невід'ємною частиною нашого життя, ми використовуємо їх для спілкування, роботи, онлайн-покупок, банківських операцій та багато іншого, на жаль, це робить їх ласою здобиччю для кіберзлочинців. Збільшення кількості та складності кібератак є важливою проблемою в сучасному цифровому середовищі. Це відбувається через постійний розвиток технологій, що надають зловмисникам нові можливості для виконання атак, а також через поширення відкритих джерел інформації про вразливості та методи атак.[16-17]

За даними Verizon, у 2023 році 41% кібератак були спрямовані на веб-додатки, що на 12% більше, ніж у 2022 році, це свідчить про зростаючу загрозу, яку представляють кіберзлочинці для цих систем. Дослідження також показують, що 73% успішних кібератак на веб-додатки використовували вразливості на фронтенді, тобто ¾ всіх атак, з цього стає зрозуміло наскільки це популярний метод викрадення даних і т.п.

Із вищесказаного випливає, що найпоширенішими типами атак на фронтенді є XSS, та SQL Injection. XSS-атаки дозволяють зловмисникам впроваджувати веб-сторінки шкідливі скрипти, які можуть використовувати конфіденційну інформацію користувачів. CSRF-атаки спонукають авторизованих користувачів виконати небажані дії на веб-сайті без їхньої згоди. SQL Injection-

атаки дозволяють зловмисникам отримувати доступ до баз даних, включаючи конфіденційну інформацію.[13, 16-17]

На жаль, історія засвідчує низку вразливостей та атак на веб-додатки. Наприклад, у 2021 році зловмисники атакували веб-сайт Shopify, вони ввели шкідливий JavaScript-код до коментаря на форумі Shopify. Коли користувачі переглядали коментар, код XSS виконувався в їхніх браузерах, даючи зловмисникам доступ до їхніх cookie, сеансових даних та інформації про замовлення. Використовуючи XSS-атаку зловмисники змогли викрасти дані про клієнтів. Також у 2022 році зловмисники атакували веб-сайт United Airlines, використовуючи CSRF-атаку, що дозволило їм бронювати квитки та змінювати маршрути польоту пасажирів без їхньої згоди.

Атака Content Injection на сайт Evernote у 2023, увівши шкідливий код до рекламного банера на сайті Evernote, коли користувачі натискали на банер, вони перенаправлялися на шкідливий веб-сайт, який завантажував шкідливе програмне забезпечення на їхні комп'ютери. У 2023 році веб-сайт Evernote став жертвою SQL Injection-атаки, що призвело до доступу зловмисників до даних користувачів. [17-19]

Також відома атака Man-in-the-Middle на сайт Newegg 2020, зловмисники створили фейкову Wi-Fi-мережу в магазині Newegg, яка виглядала як легітимна. Коли користувачі підключалися до цієї мережі, зловмисники перехоплювали їхній трафік і крали їхні дані платіжних карток. Крім цього атака SQL Injection на сайт LinkedIn у 2016 показала черговий раз наскільки вразливий фронтенд. Зловмисники ввели шкідливий SQL-код до форми входу на LinkedIn, а коли користувачі вводили свої дані для входу, код SQL Injection виконувався на веб-сервері, даючи зловмисникам доступ до даних бази даних LinkedIn, включаючи паролі користувачів.

Статистика засвідчує зростання кількості кібератак на фронтенді протягом останніх років. За даними звіту «Web Application Threat Report 2023» від компанії Imperva, кількість атак на веб-додатки зросла на 24% порівняно з попереднім

роком, це все демонструє зростаючу зацікавленість зловмисників у вразливостях, пов'язаних із фронтендом. [17-19]

Варто також розглянути статистику щодо країн які відповідальні за найбільшу кількість кібер атак, хоч саме ця діаграма показує не конкретні дані про фронтенд, все ж важливо розуміти об’єми загальних атак.

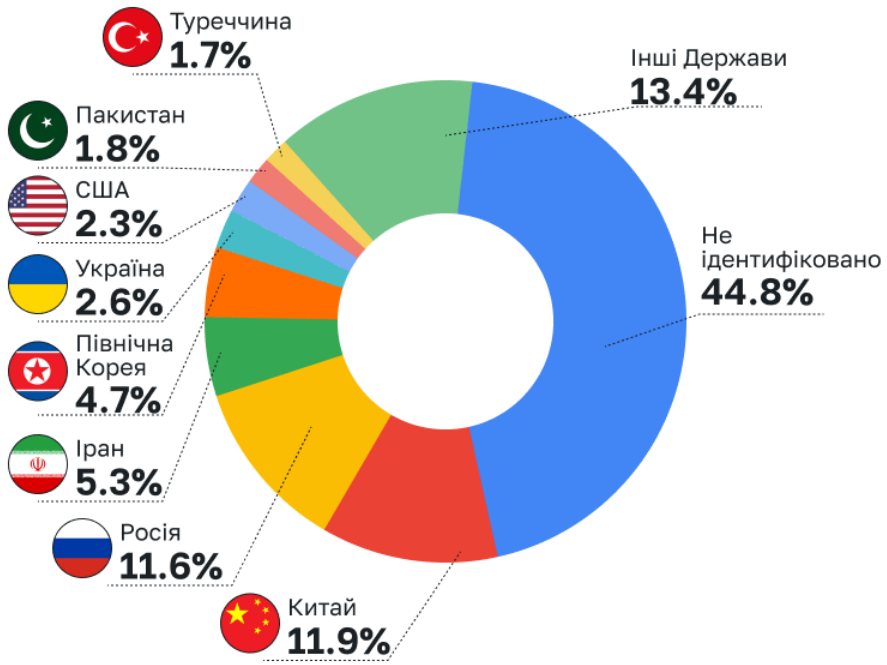

Рис. 1.6. Країни, що відповідальні за значну частину кібер-інцидентів з політичною метою з 2000 по 2023 рік.[26]

Однією з причин зростання кількості кібератак на фронтенді є поява більш складних та витончених методів атаки. Наприклад, атаки типу (XSS) та (CSRF) стають більш розповсюдженими та складними для виявлення та запобігання. Одним із відомих прикладів складних кібератак на фронтенді є атаки через маніпулювання вмістом клієнтського боку (Client-Side Content Manipulation). Зловмисники використовують скрипти для зміни вмісту веб-сторінок під час їх

завантаження, що частіше за все призводить до витоку конфіденційної інформації або зловживання довіри користувачів.

2. **Проактивний захист від кібератак на фронтенді**

Постійна увага кібербезпеці під час розробки інтерфейсу має вирішальне значення, оскільки вона стає визначальним фактором у забезпеченні безпеки веб-додатків. Розробники повинні бути пильними, уникати таких поширених вразливостей, як міжсайтовий сценарій (XSS) наприклад, і гарантувати захист конфіденційності даних користувачів. Наукові дослідження та практичний досвід підтверджують, що виробництво безпечних веб-програм можливе завдяки систематичній і прорахованій роботі з аспектами кібербезпеки на етапі проектування та розробки, тобто проактивного захисту. [20]

Застосування проактивних заходів безпеки у розробці інтерфейсу є критичним з міркувань не лише захисту конфіденційності, а й забезпечення надійності та авторитету веб-додатків. Дослідження з безпеки веб-додатків підтверджують, що більшість кібератак спрямовані саме на фронтенд, тобто на інтерфейс, з яким взаємодіє користувач. Тому, правильне планування та виконання заходів забезпечення кібербезпеки на етапі розробки є критично важливим для запобігання можливих атак.

Проактивні заходи безпеки є критичними на ранніх етапах розробки веб-додатків. Вони дозволяють виявити та виправити потенційні вразливості у коді ще до того, як вони можуть бути використані для атак, таких як XSS або інші атаки на фронтенд, що забезпечує економію ресурсів, оскільки вирішення проблем на ранніх етапах коштує менше, ніж пізніше виявлення та виправлення їх після випуску додатку. Забезпечення високого рівня кібербезпеки з самого початку робить веб-додатки більш привабливими для користувачів, оскільки вони можуть бути впевнені в безпеці своїх даних. Крім того, проактивні заходи допомагають уникнути серйозних проблем та інцидентів безпеки, що можуть призвести до втрати часу та коштів на відновлення даних та відновлення репутації

компанії. Не менш важливо, що вони допомагають відповідати вимогам стандартів безпеки, таких як PCI DSS або GDPR, та уникнути санкцій за порушення цих вимог.[20]

Щоб досягти успіху у проактивному захисті від кібератак на фронтенді, розробники повинні дотримуватися найкращих практик безпеки програмування та використовувати передові техніки захисту. Наприклад, застосування автоматизованих інструментів виявлення вразливостей у коді, регулярне проведення аудитів безпеки та впровадження систем моніторингу можливих загроз допомагають забезпечити ефективний рівень безпеки. [20]

2.1. Переваги проактивного захисту

Проактивний захист - це підхід до кібербезпеки, який фокусується на запобіганні кібератакам, а не на їх реагуванні. Це включає виявлення та усунення вразливостей у веб-додатку до того, як ними зможуть скористатися зловмисники.

Існує багато переваг використання проактивного підходу до кібербезпеки на фронтенді, тобто виявлення та усунення вразливостей на ранніх етапах розробки допомагає зменшити ризик успішних кібератак, а також захищає конфіденційні дані користувачів від несанкціонованого доступу. Крім того, активне впровадження проактивних заходів безпеки вказує на серйозне ставлення до кібербезпеки, що сприяє підвищенню репутації бренду та зменшенню фінансових втрат у разі кібератак. Кібератаки можуть негативно впливати на досвід користувачів та викликати перебої в роботі систем.

Проактивний захист допомагає забезпечити безперебійну та безпечну роботу веб-додатків, що підвищує задоволеність користувачів та їх продуктивність. У сучасному бізнес-середовищі, де кібербезпека відіграє ключову роль, проактивний захист може надати конкурентну перевагу, а також допомагає відповісти на нормативні вимоги щодо безпеки даних та уникнути штрафів. Також, впровадження проактивних заходів безпеки сприяє формуванню культури

кібербезпеки в організації, де всі співробітники розуміють важливість захисту від кібератак та активно беруть участь у забезпеченні безпеки веб-додатків.

Нарешті, знання, що всі можливі заходи безпеки були прийняті, надає керівництву та співробітникам душевний спокій та впевненість у захищеності їх систем від кіберзагроз.

Впровадження засобів автоматизації в кіберзахисті має вирішальне значення у зниженні часу визначення подій безпеки та формування відповіді на потенційні вторгнення. Однією з основних переваг автоматизованих систем є їх здатність реагувати на загрози в реальному часі та проводити аналіз великого обсягу даних, що перевищує можливості людського персоналу. Наприклад, системи машинного навчання та штучного інтелекту можуть виявляти незвичайну активність або аномалії в мережі, що може бути ознакою можливого кібератаки. Такі системи можуть автоматично спрямовувати інформацію про потенційні загрози або навіть ініціювати заходи для їх припинення.

Більше того, автоматизовані системи можуть допомогти в управлінні відповіддю на кібератаки, допомагаючи при реалізації стратегії інцидентного реагування. Наприклад, автоматичне розгортання захисних заходів або блокування доступу до певних ресурсів у разі виявлення кіберзагрози може значно зменшити час реакції та мінімізувати можливі збитки.

Однак важливо зауважити, що автоматизація повинна бути доповнена ефективними процедурами та стратегіями кіберзахисту. Недостатньо лише впровадити автоматичні системи, не розуміючи специфіку кіберзагроз та не встановлюючи належних заходів безпеки. Проактивний кіберзахист вимагає поєднання технологій, процесів та експертного знання для ефективного виявлення, запобігання та реагування на кіберзагрози.

2.2. Методи проактивного захисту

Для боротьби із загрозою сучасних кібератак потрібне сучасне рішення, яке дозволяє організаціям адаптуватися до можливостей у режимі реального часу.

Серед проактивного захисту можна виділити аудит безпеки коду - це систематичне перевірка вихідного коду веб-додатка на наявність потенційних вразливостей та слабкі місця, які можуть бути використані зловмисниками для атаки. Цей процес включає в себе ретельний аналіз програмного коду з метою виявлення можливих вразливостей безпеки та вирішення їх, щоб запобігти можливим кібератакам. На початковому етапі визначається обсяг і завдання аудиту безпеки, тому важливо визначити, які саме аспекти безпеки будуть перевірятися, які інструменти та методи будуть використані для аналізу коду, а також скласти план дій та розподілити завдання між аудиторами.

Після планування проводиться аналіз вихідного коду веб-додатка. Аудитори ретельно переглядають код кожного компонента додатка з огляду на можливі вразливості, такі як XSS, SQL-ін'єкції, недоліки аутентифікації та авторизації, недостатні контролі на введення даних, використання застарілих або небезпечних функцій тощо. Потім, під час аналізу вихідного коду аудитори активно шукають потенційні вразливості та слабкі місця в безпеці, для цього вони можуть використовувати різноманітні інструменти автоматичного аналізу коду, а також проводити ручний аналіз та тестування з використанням різних сценаріїв атаки.

Після виявлення потенційних вразливостей в коді вони оцінюються з точки зору потенційного впливу на систему та ймовірності їх використання зловмисниками. Вразливості класифікуються за серйозністю та пріоритетом для виправлення. Після завершення аналізу аудитори складають детальний звіт, в якому фіксуються всі виявлені вразливості, їх опис, серйозність та рекомендації щодо їх виправлення. Цей звіт надається розробникам для подальшої роботи над виправленням виявлених проблем.

Впровадження захисту від міжсайтового скриптингу (XSS) також є важливою складовою проактивного захисту фронтенду веб-додатків, установка правильних заголовків безпеки, таких як Content-Security-Policy (CSP), дозволяє обмежити виконання скриптів на сторінці та уникнути XSS-атак.

Усі дані, що надходять від користувачів, повинні бути екрановані та фільтровані перед тим, як вони використовуються у веб-додатку. Це допомагає уникнути виконання шкідливих скриптів, які можуть бути вставлені користувачами. Використання безпечних API, таких як Document Object Model (DOM) API, дозволяє маніпулювати HTML та XML документами без ризику виконання XSS-атак. [7, 19]

Регулярне оновлення використовуваних бібліотек та фреймворків дозволяє уникнути використання відомих вразливостей XSS, оскільки вони часто виправляються у нових версіях. Проведення регулярних тестів на XSS-вразливості дозволяє виявити та виправити потенційні проблеми безпеки на ранніх етапах розробки.

Захист від вразливостей, що виникають при використанні контенту з інших доменів (Cross-Origin Resource Sharing - CORS), є важливою складовою безпеки веб-додатків. CORS визначає, як веб-браузери повинні поводитися при обміні ресурсами між різними доменами. Недоліки в конфігурації CORS можуть призвести до витоку конфіденційних даних, збільшити ризик XSS-атак та інших безпекових загроз. Використання правильних налаштувань CORS, таких як налаштування заголовків Access-Control-Allow-Origin, дозволяє обмежити доступ до ресурсів лише з визначених доменів, що допомагає уникнути несанкціонованого доступу до конфіденційних даних. [22]

Для забезпечення додаткового рівня безпеки можна використовувати механізми авторизації та аутентифікації, такі як токени доступу або сеансові куки, щоб перевіряти права доступу до ресурсів, навіть якщо запити CORS відправляються з надійних доменів. Перевірка та валідація всіх запитів, які надходять з інших доменів через CORS, допомагає уникнути виконання шкідливих або несправних запитів, що може призвести до викрадення або зміни даних. Проведення регулярних тестів на вразливості CORS дозволяє виявляти та виправляти потенційні проблеми безпеки на ранніх етапах розробки та впровадження. Постійний моніторинг запитів CORS та журналювання подій дозволяє вчасно виявляти та реагувати на потенційні загрози та атаки.

Використання захищених протоколів зв'язку є критичним для забезпечення безпеки веб-додатків. HTTPS, TLS та SSL - це основні протоколи, які забезпечують конфіденційність, цілісність та автентичність передачі даних між клієнтом і сервером, запобігаючи їх перехопленню, зміні або підробленню зловмисниками. Перевірка сертифікатів SSL/TLS та регулярне оновлення конфігурацій сервера та криптографічних бібліотек є ключовими аспектами забезпечення безпеки протоколів зв'язку.[22]

Постійний моніторинг безпеки серверів і виявлення вразливостей в захищених протоколах зв'язку допомагає вчасно виявляти та виправляти потенційні загрози безпеки.

2.3. Навчання та обізнаність

Люди становлять найбільш вразливу ланку у цифровому світі. Це може виявитися миттєво: працівник недбало відкриває електронний лист, і всю компанію чекають серйозні проблеми. Ці наслідки можуть бути надзвичайно важливими: зашифровані IT-системи, вимоги викупу, недозволені маніпуляції з даними та зупинення нормального ходу процесів. [23] Навчання розробників та адміністраторів про актуальні загрози XSS та найкращі практики безпеки допомагає забезпечити своєчасну реакцію на можливі загрози.

Способи підвищення обізнаності про кібератаки на фронтенд.

Спосіб	**Опис**	**Переваги**
Проведення навчальних курсів	Допомога розробникам та власникам веб-додатків зрозуміти ризики та методи кібератак.	Глибоке занурення у тему, можливість отримати відповіді на питання від експертів.

Читання статей та блогів	Доступ до інформації про кібератаки на фронтенд з різних джерел.	Безкоштовний та зручний спосіб дізнатися про актуальні загрози та методи захисту.
Відвідування конференцій та семінарів	Ознайомлення з останніми кіберзагрозами та методами захисту від фахівців.	Можливість спілкуватися з експертами та іншими учасниками, дізнатися про нові інструменти та ресурси.
Використання інструментів сканування	Виявлення вразливостей у веб-додатках.	Дозволяє швидко та легко знайти потенційні проблеми з безпекою.
Впровадження програм навчання	Підвищення обізнаності про кібербезпеку серед усіх співробітників.	Допомагає створити культуру кібербезпеки в організації, де всі розуміють важливість захисту даних.

Таблиця 2.1

Навчання та обізнаність - це не лише інструменти забезпечення безпеки, але й інвестиція в майбутнє, яка може запобігти серйозним проблемам та втратам в майбутньому.

Існує багато способів підвищити обізнаність про кібератаки на фронтенд. Деякі з них включають:

- проведення навчальних курсів. (Навчальні курси можуть допомогти розробникам та власникам веб-додатків зрозуміти ризики та методи кібератак.)
- читання статей та блогів. (Існує багато статей та блогів, які надають інформацію про кібератаки на фронтенд.)

- відвідування конференцій та семінарів. (Конференції та семінари можуть бути чудовим способом дізнатися про останні кіберзагрози та методи захисту.)
- використання інструментів сканування. (Існує багато інструментів сканування, які можуть допомогти виявити вразливості у веб-додатках.)
- впровадження програм навчання. (Програми навчання можуть допомогти підвищити обізнаність про кібербезпеку серед усіх співробітників.)

Навчання та обізнаність про кібератаки на фронтенд є важливою частиною будь-якої стратегії кібербезпеки. Коли розробники та власники веб-додатків розуміють ризики та методи кібератак, вони можуть вжити заходів для захисту своїх веб-додатків та їх користувачів.

3. РОЗРОБКА ЗАСТОСУНКУ З ПРОАКТИВНИМ ЗАХИСТОМ НА ФРОНТЕНДІ

У попередніх розділах ми розглянули теоретичні аспекти кібератак на фронтенд веб-застосунків та важливість проактивного захисту. На основі отриманих знань, у цьому розділі буде описано процес розробки практичного застосунку, в якому інтегровані методи проактивного захисту від кіберзагроз.

Метою цього розділу є надати детальне пояснення кроків, пов’язаних зі створенням захищеного веб-застосунку, від вибору технологій та інструментів до реалізації та тестування функцій безпеки. Він зосереджений на поясненні структури додатка, методах впровадження проактивних заходів безпеки та аналізі результатів тестування вразливостей. Створення цієї програми демонструє, як теоретичні знання застосовуються на практиці, і пропонує розробникам корисні поради щодо захисту зовнішніх веб-програм, зниження ризиків кібербезпеки та підвищення довіри користувачів.

Предметною областю для застосунку став сервіс для онлайн голосувань. Пошук теми базувався на актуальності та необхідності у проактивному захисті. В такому виді застосунків, уся влада є в руках користувачів, оскільки вони є основними наповнювачами веб сторінки. Це означає, що розробнику потрібно приділити особливу увагу безпеці застосунку, щоб користувачі сайту не стали жертвами зловмисників, які матимуть на меті заволодіти чужими даними.

3.1. Вибір технологій та інструментів

Під час вибору інструментів для розробки особлива увага приділялась не тільки мінімізації потенційних вразливостей в технології, а також в актуальності та підтримці технологій, постійному пошуку та усуненню можливих слабких місць у вихідному коді. Вибір пав на Angular як клієнтську сторону та Firebase як на серверну частину.

3.1.1. Firebase

Google Firebase є однією з найпопулярніших платформ для розробки веб- і мобільних застосунків, яка надає широкий спектр інструментів для створення серверної інфраструктури. Однією з ключових переваг Firebase є його легкість у використанні та інтеграції з іншими сервісами Google. Firebase забезпечує розробників такими функціями, як автентифікація користувачів, зберігання даних у реальному часі, хмарні функції, що дозволяє швидко і ефективно будувати масштабовані і надійні серверні рішення. Однією з важливих особливостей Firebase є його потужна система автентифікації, яка підтримує різні методи входу, включаючи email і пароль, телефон, а також авторизацію через популярні соціальні мережі. Це значно полегшує процес управління користувачами та підвищує загальний рівень безпеки застосунку. Крім того, Firebase надає можливість реалізації двофакторної автентифікації, що є важливим елементом проактивного захисту від несанкціонованого доступу.[28]

Функція зберігання даних у реальному часі Firebase Realtime Database і Firestore дозволяють зберігати та синхронізувати дані між користувачами в режимі реального часу. Це особливо корисно для застосунків, які вимагають миттєвого оновлення даних, таких як чати, онлайн-ігри або колаборативні платформи. Вбудовані механізми безпеки, такі як правила безпеки Firebase, дозволяють контролювати доступ до даних і забезпечувати їх захист на рівні бази даних.Хмарні функції Firebase Functions дозволяють виконувати серверний код у відповідь на події, що відбуваються у вашому застосунку. Це дозволяє створювати складну логіку без необхідності управління серверами, що значно спрощує процес розробки та підтримки застосунку. За допомогою хмарних функцій можна реалізовувати такі проактивні заходи безпеки, як моніторинг активності користувачів, виявлення підозрілих дій та автоматичне реагування на потенційні загрози.[28]

3.1.2. Angular

Angular 17 є потужним фронтенд-фреймворком від Google, який використовується для створення динамічних і високопродуктивних веб-застосунків. Angular надає широкий набір інструментів і бібліотек, що дозволяють розробникам ефективно будувати складні інтерфейси користувача з високою продуктивністю та безпекою. Одна з основних переваг Angular полягає у його модульній архітектурі, що дозволяє легко розділяти застосунок на незалежні компоненти, які можуть бути повторно використані та протестовані окремо.[26]

Angular включає потужні засоби для роботи з формами та валідацією даних, що дозволяє забезпечити високий рівень захисту від типових фронтенд-атак, таких як XSS (міжсайтовий скриптинг). Завдяки двосторонньому зв'язуванню даних, Angular дозволяє миттєво оновлювати інтерфейс користувача у відповідь на зміни у моделях даних, що робить його ідеальним вибором для розробки інтерактивних веб-застосунків.

Однією з ключових функцій Angular є його потужний інструмент для роботи з сервісами та залежностями — Angular Dependency Injection (DI). DI дозволяє легко управляти залежностями між компонентами, що сприяє більш структурованій та організованій розробці. Це також сприяє підвищенню безпеки, оскільки дозволяє більш ефективно управляти доступом до критично важливих ресурсів застосунку. [26]

Завдяки вбудованим засобам для управління маршрутизацією, Angular дозволяє створювати багатосторінкові застосунки з плавним переходом між сторінками. Це дозволяє реалізувати складні сценарії навігації та підвищити загальний користувацький досвід. Крім того, Angular підтримує Lazy Loading (відкладене завантаження), що дозволяє завантажувати модулі тільки тоді, коли вони необхідні, зменшуючи час завантаження застосунку та підвищуючи його продуктивність.

Нарешті, Angular CLI (Command Line Interface) надає потужний інструментарій для автоматизації процесу розробки, включаючи генерацію компонентів, сервісів та інших елементів застосунку. CLI також підтримує інтеграцію з популярними інструментами тестування, такими як Jasmine і Karma, що дозволяє забезпечити високу якість та надійність коду. Це робить Angular ідеальним вибором для розробки масштабованих та безпечних веб-застосунків.[26]

3.2. Архітектура застосунку

Оскільки тема наукової роботи присвячена проактивному захисту на фронтенді, то і основний акцент був зроблений на клієнтську частину.

Застосунок можна умовно поділити на 4 ключових компонентів.

Сторінка авторизації – перше що бачить користувач і потенційний злочинець коли потрапляє на сайт. Вона поділений на три незалежні компоненти, такі як: компонент авторизації, реєстрації та компонент відновлення паролю.

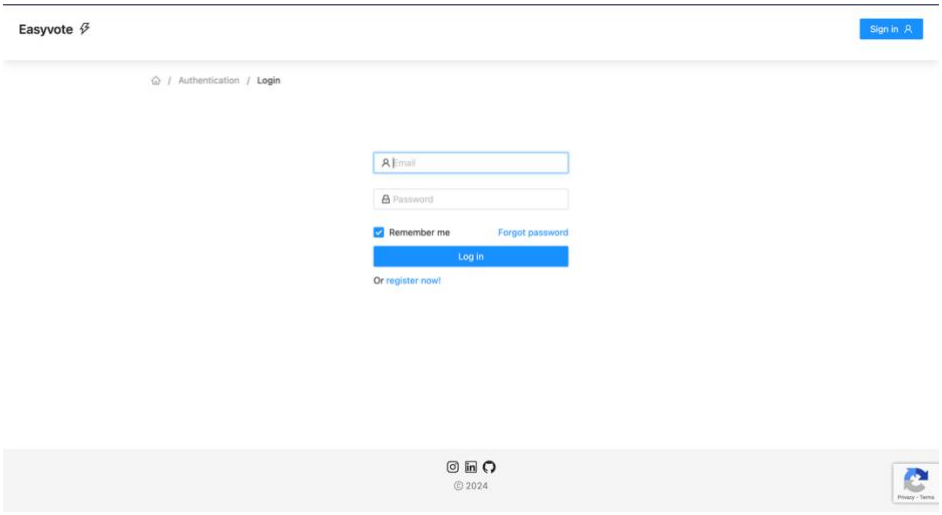

Рис 3.1 – сторінка авторизації

Головна сторінка – наступне що бачить користувач після успішної реєстрації. Ця сторінка містить список усіх активних та неактивних опитувань.

Якщо натиснути на назву опитування, можна перейти на третій компонент - це сторінка самого опитування. Там можна побачити повний опис та можливі варіанти для голосування.

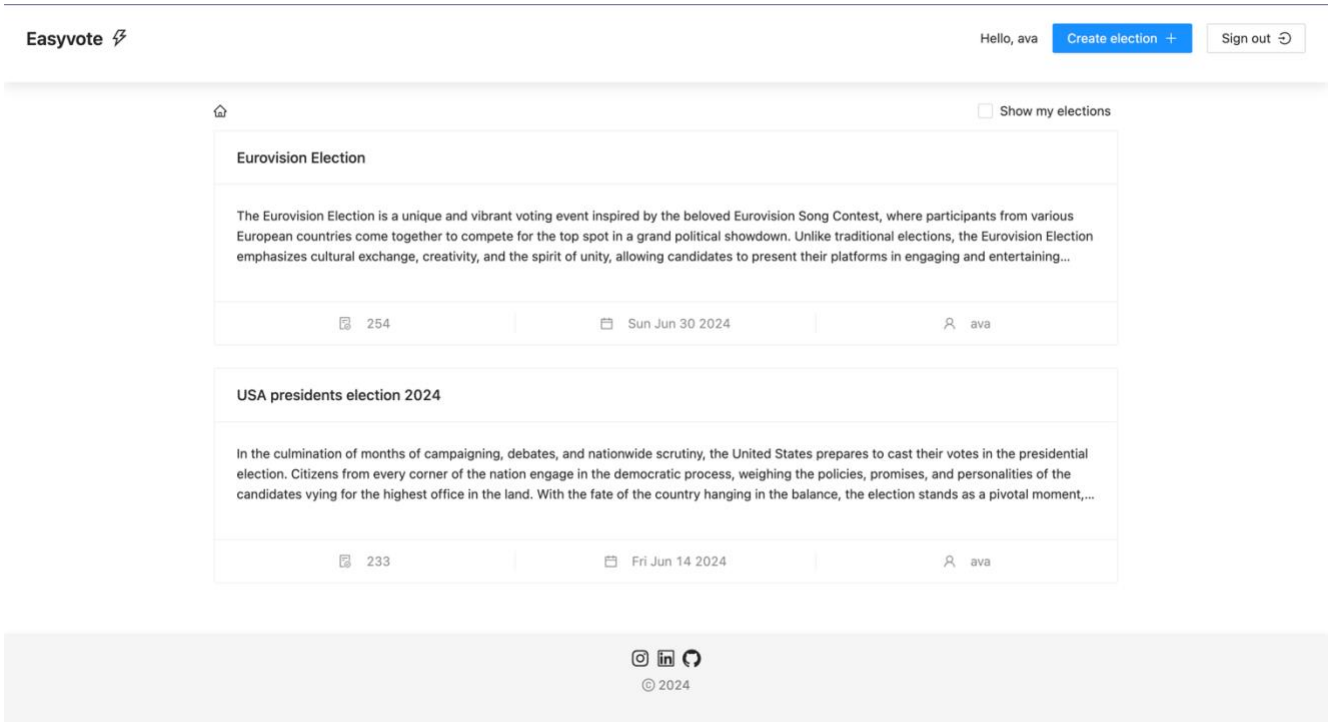

Рис 3.2 – головна сторінка

Останній ключовий компонент – форма створення опитування. В ній можна задати назву, опис, дату початку та кінця, та можливі опції для голосування. Вона відіграє велику роль в даному застосунку, оскільки саме через цей компонент потенційний зловмисник може намагатись заподіяти шкоду користувачам.

Таким чином, маємо наступну діаграму діяльності користувача в застосунку:

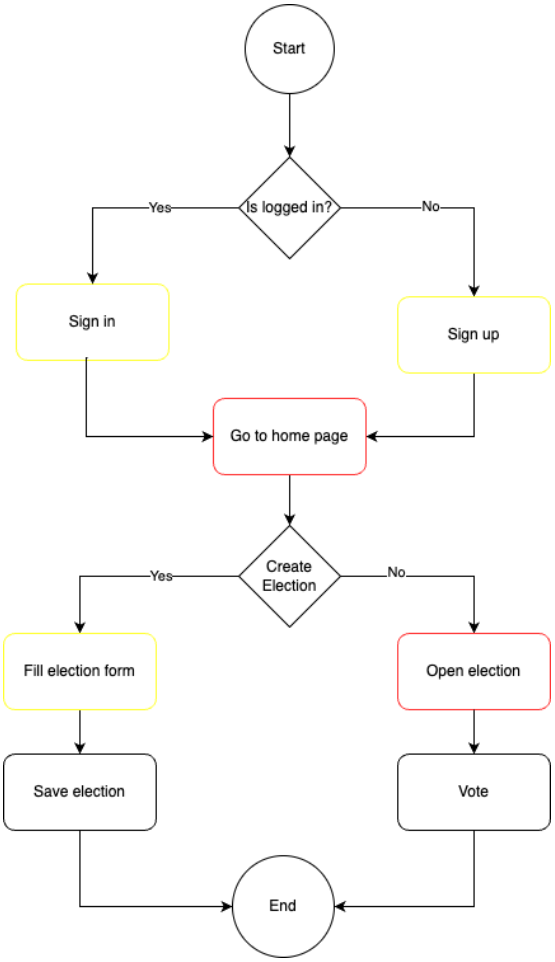

Рис. 3.3 діаграма діяльності користувача

Жовтим кольором обведені діяльності, в яких користувач повинен вводити певні дані, які будуть відображені в застосунку для інших. Це є потенційно небезпечні місця, з котрих ймовірніше за все зловмисник буде намагатись почати атаку. Червоним позначені місця в котрих буде відображено всю ту інформацію, яку заповнюють користувачі. Так наприклад, якщо зловмисник буде планувати XSS атаку, він буде намагатись створити таке опитування, яке буде місити певний скрипт, котрий буде запускатись щоразу коли інший користувач буде відкривати дану сторінку.

3.3. Проактивні методи захисту

3.3.1. Аутентифікація та авторизація

У рамках реалізації аутентифікації та авторизації у веб-застосунку було використано сервіс Google Firebase. Firebase Authentication надає потужний та гнучкий інструментарій для інтеграції різних методів входу, включаючи електронну пошту та пароль, а також OAuth-провайдери, такі як Google, Facebook, Github та інші. Це забезпечує високу гнучкість і зручність для користувачів, які можуть вибирати найбільш зручний для них спосіб аутентифікації.

```
1+ usages  new *
googleSignIn(): Observable<void> {
    const provider : GoogleAuthProvider  = new GoogleAuthProvider();
    return from(signInWithPopup(this.firebaseAuth, provider)).pipe(switchMap( project: () => of( value: void 0)));
}
1+ usages  new *
gitSignIn(): Observable<void> {
    const provider : GithubAuthProvider  = new GithubAuthProvider();
    return from(signInWithPopup(this.firebaseAuth, provider)).pipe(switchMap( project: () => of( value: void 0)));
}
```

Рис. 3.4 - Авторизація за допомогою OAuth-провайдерів Google та Github

Однією з головних переваг Firebase Authentication над іншими методами аутентифікації є його інтеграція з різними платформами, включаючи веб, Android та iOS. Це забезпечує простоту впровадження та використання єдиного рішення для аутентифікації на всіх платформах. Крім того, Firebase забезпечує високий рівень безпеки завдяки використанню сучасних методів шифрування та захисту даних. Сервіс сертифікований за стандартами ISO/IEC 27001, SOC 1, SOC 2 та SOC 3, що гарантує відповідність високим стандартам безпеки.[28]

```
register(email: string, password: string, username: string): Observable<void> {
    const promise : Promise<void>  = createUserWithEmailAndPassword(
        this.firebaseAuth,
        email,
        password
    ).catch((err) : void  => {
        throw err;
    }).then(response : UserCredential  => {
        return updateProfile(response.user, {displayName, photoURL: photoUrl}: {displayName: username})
    })
    return from(promise);
}
```

Рис. 3.5 – запит на реєстрацію використовуючи введені дані користувача

Для зберігання сесійних даних користувачів використовуються JWT, які забезпечують безпечне зберігання та передачу інформації. Токени шифруються та підписуються, що гарантує їх цілісність і захист від підробки. Важливим аспектом безпеки є обмежений термін дії токенів, що мінімізує ризики, пов'язані з їх викраденням. Firebase також використовує захищені протоколи передачі даних HTTPS, що забезпечує захист від атак типу MITM.[28]

Одним з важливих аспектів зберігання даних користувачів є використання IndexedDB, що забезпечує значні переваги над традиційними Cookies. IndexedDB є низькорівневим API для зберігання значних обсягів структурованих даних у браузері користувача. На відміну від Cookies, які обмежені в обсязі та часто передаються разом з кожним HTTP-запитом, IndexedDB дозволяє зберігати більші обсяги даних локально без впливу на продуктивність мережевих запитів.

З точки зору безпеки, IndexedDB надає кращий контроль над доступом до даних. Доступ до IndexedDB можливий лише з того ж домену, що і застосунок, що запобігає міжсайтовим атакам. Крім того, дані в IndexedDB не відправляються автоматично з кожним запитом на сервер, що значно знижує ризик їх викрадення під час передачі. Використання IndexedDB також дозволяє розробникам впроваджувати більш складні механізми шифрування та управління доступом до даних, що забезпечує додатковий рівень захисту.

Таким чином, використання Google Firebase у поєднанні з IndexedDB забезпечує надійну та безпечну аутентифікацію та авторизацію, яка відповідає сучасним вимогам безпеки і дозволяє створити зручний та масштабований веб-застосунок.[28]

3.3.2. Захист від XSS атак

3.3.2.1. Валідація даних

Для захисту від XSS атак перше, що необхідно впровадити – це сильну валідацію для усіх можливих полів, котрі має змогу заповнювати користувач. Для впровадження валідацій необхідно написати регулярний вираз. Якщо написаний користувачем текст не відповідає виразу, користувач не зможе зберегти ним текст, котрий потенційно може містити скрипт.[12]

```
const regex = /<\s*script\b[^>]*>(.*?)<\s*\/\s*script\s*>|\
<\s*(iframe|embed|object|applet|meta|link|style|base|form|input|button|textarea|sel
\b(on\w+)\s*=\s*['"]?[^'"]*['"]?|\
<\s*a\s+[^>]*href\s*=\s*['"]?\s*javascript:[^'"]*['"]?\s*>/gi;
```

Рис. 3.6 – приклад регулярного виразу проти XSS атак

Цей регулярний вираз призначений для пошуку та ідентифікації потенційно шкідливих або небажаних HTML-елементів і атрибутів у рядку. Він призначений, щоб реагувати на будь які потенційно небезпечні html елементи, атрибути, об’єкти. Так наприклад, якщо спробувати вставити <img src=”example.png” onerror=”console.log(‘Potential vulnerability’)”> то регулярний вираз буде сигналізувати що скрипт, котрий може виконитась.

```
form: FormGroup<{
    email: FormControl<string>
    password: FormControl<string>;
    remember: FormControl<boolean>;
}> = this.fb.group( controls: {
    email: ['', [Validators.required, Validators.pattern( pattern: /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/)]],
    password: ['', [Validators.required, Validators.pattern( pattern: /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&_])[A-Za-z\d@$!%*?&_]{8,}$/)]],
    remember: [true]
});
```

Рис. 3.7 – Використання регулярних виразів для валідації полів

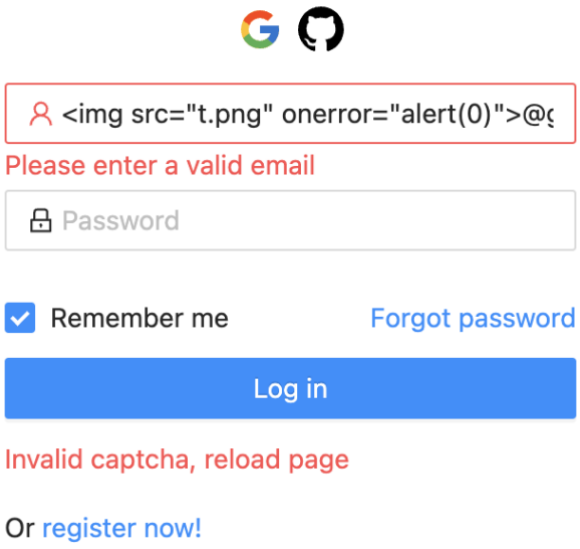

Рис. 3.8 – Результат відпрацювання регулярного виразу

У застосунку було впроваджено цілий ряд валідацій для полів котрі заповнюють користувачі. Це дозволило унеможливити XSS атаку через поля вводу.

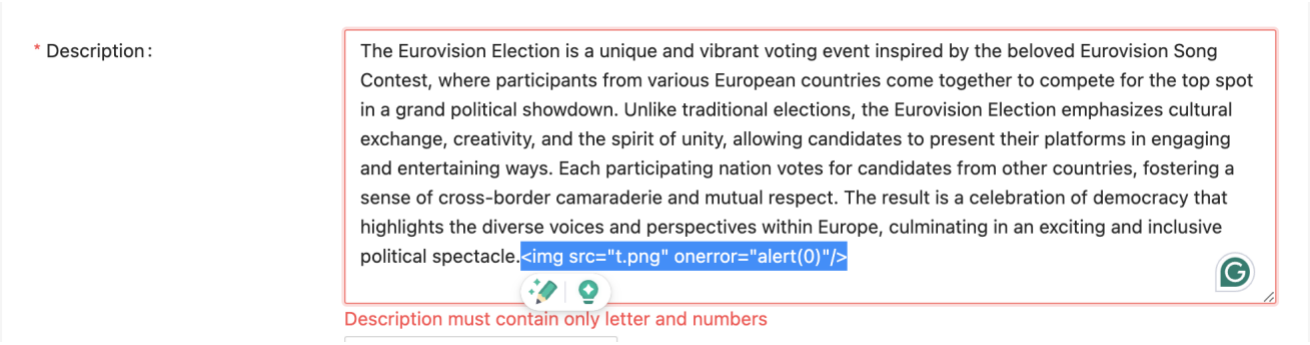

Рис. 3.8.1 – Результат відпрацювання регулярного виразу

На малюнку 3.8.1 можна побачити що при спробі додати до опису тег *img* з атрибутом *onerror*, котрий в разі якщо не знається картинка t.png викличе метод *alert(0)*, відпрацював валідатор і повідомив що допустимо тільки літери старшого та молодшого регістрів і цифри.

Сам регулярний вираз виглядає наступним чином: */^[a-zA-Z0-9][a-zA-Z0-9]*[a-zA-Z0-9]$/* І використовується для перевірки рядків, які:

- Починаються з буквено-цифрового символу.
- Закінчуються буквено-цифровим символом.
- Містять лише алфавітно-цифрові символи та пробіли між ними.

3.3.2.2. Content Security Policy

CSP - це потужний інструмент для запобігання XSS атакам. Він дозволяє визначити, які джерела контенту дозволені для виконання в веб-застосунку. Це обмежує можливості виконання зловмисного коду, навіть якщо він був впроваджений у веб-сторінку. CSP також дозволяє блокувати виконання inline-скриптів та обмежувати використання eval().[]

Застосунок використовує наступну політику CSP:

- default-src 'self';
- script-src 'self' https://vote-app-2b5ab.web.app https://apis.google.com https://www.gstatic.com https://www.google.com 'unsafe-inline';
- style-src 'self' https://vote-app-2b5ab.web.app 'unsafe-inline';
- img-src 'self' data: https://vote-app-2b5ab.web.app;
- font-src 'self' https://vote-app-2b5ab.web.app;
- connect-src 'self' https://firestore.googleapis.com https://identitytoolkit.googleapis.com;
- frame-ancestors 'self';
- frame-src 'self' https://www.google.com https://vote-app-2b5ab.firebaseapp.com;
- object-src 'none';

- base-uri 'self';
- form-action 'self';

Пояснення політики:

default-src 'self' - Дозволяє завантаження контенту тільки з того ж домену.

script-src.- Дозволяє виконання скриптів тільки з власного домену та певних надійних джерел, таких як Google та Firebase. Використання 'unsafe-inline' у даному випадку є вимушеною мірою, оскільки Firebase не надає змоги генерувати nonce фразу.

style-src - Дозволяє завантаження стилів з власного домену та зазначених джерел. 'unsafe-inline' дозволяє вбудовані стилі, що також може бути ризикованим.

img-src - Дозволяє завантаження зображень з власного домену, зазначених джерел та даних URI.

font-src - Дозволяє завантаження шрифтів з власного домену та зазначених джерел.

connect-src: Дозволяє з'єднання з власного домену та певних зовнішніх API, таких як Firestore та Google Identity Toolkit.

frame-ancestors 'self' - Дозволяє вбудовування сторінок лише у фрейми з того ж домену.

frame-src - Дозволяє вбудовування фреймів лише з зазначених джерел.

object-src 'none' - Забороняє використання тегів <object>, <embed> та <applet>.

base-uri 'self' - Обмежує основний URI до того ж домену.

form-action 'self' - Обмежує відправку форм лише на той же домен.

3.3.3. Вбудований захист Angular

Для захисту від XSS атак Angular позначає усі значення як неперевірені. Коли це значення вставляється в DOM браузера через innerHTML, або через інтерполяцію({{}}), фреймворк дезинфікує та екранує це значення. Це означає, що будь-який текст, який вставляється у HTML-документ через Angular-шаблони, буде безпечним для відображення і не виконається як скрипт. Це значно знижує ризик XSS атак, коли зловмисник намагається впровадити шкідливий скрипт через користувацький ввід.[26]

Дезинфекція - це перевірка ненадійного значення, яка перетворює його на значення, яке можна безпечно вставляти в DOM. У багатьох випадках санітарна обробка взагалі не змінює значення. Санація залежить від контексту: Значення, безпечне в CSS, може бути потенційно небезпечним в URL-адресі.

Рис. 3.9 – попередження про дезінфекцію небезпечного скрипта

Спробувавши вставити виконуваний код в DOM, Angular повідомив в консолі браузера про те що він утилізував небезпечний скрипт, заливши тільки тег *img*.

Angular також надає методи для явно вказувати, що певний контент є безпечним для вставки у DOM, за допомогою класу DomSanitizer. Ці методи включають

- BypassSecurityTrustHtml
- BypassSecurityTrustScript
- BypassSecurityTrustStyle
- BypassSecurityTrustUrl
- BypassSecurityTrustResourceUrl.

Використання цих методів слід обмежити і застосовувати тільки після ретельної перевірки вхідних даних, оскільки вони обходять вбудовані механізми захисту

Angular. У нашому випадку вийшло обійтись використанням даного класу, оскільки не було потреби у прямому додаванню довірених елементів.

```
constructor(private sanitizer: DomSanitizer) {
  // javascript: URLs are dangerous if attacker controlled.
  // Angular sanitizes them in data binding, but you can
  // explicitly tell Angular to trust this value:
  this.dangerousUrl = 'javascript:alert("Hi there")';
  this.trustedUrl = sanitizer.bypassSecurityTrustUrl(this.dangerousUrl);
```

Рис. 4.0 – Приклад використання DomSanitizer

Angular забезпечує механізми маршрутизації, а саме Guard-и, які дозволяють контролювати доступ до різних маршрутів у застосунку. Методи CanActivate, canDeactivate, Resolve, CamLoad та інші можуть використовуватись для перевірки автентифікації та авторизації користувача перед тим як надати доступ до певних частин застосунку. [26]

Так наприклад у веб застосунку було використано метод CanActivate, котрий виконується кожного разу, коли користувач намагається перейти на сторінку редагування опитування. У разі якщо користувач не авторизований, Guard поверне його на сторінку авторизації

```
{
    path: 'creation/:id',
    loadComponent: () => import('./features/elections-management/election-creation-form/election-creation-form.component'),
    canActivate: [authGuard]
},
```

Рис. 4.1 – використання Angular Guard CanActivate

Вбудовані механізми захисту Angular забезпечують високий рівень безпеки від більшості типів атак, включаючи XSS, CSRF, та інші поширені вразливості. Однак, жоден фреймворк не може повністю гарантувати захист від усіх можливих атак. Важливо розуміти, що безпека веб-застосунку залежить від комплексного

підходу, який включає належну конфігурацію серверу, використання захищених протоколів (HTTPS), налаштування правильних HTTP-заголовків безпеки, таких як CSP, та регулярне тестування безпеки.[26]

3.3.3.1. Google ReCaptcha

Одним з ефективних методів захисту веб-застосунків від автоматизованих атак є використання Google reCAPTCHA. Це сервіс, який допомагає відрізняти реальних користувачів від ботів, забезпечуючи додатковий рівень захисту для форм вводу, реєстрації, входу в систему та інших важливих взаємодій у веб-застосунку.[29]

Переваги використання Google reCAPTCHA

Захист від ботів:

Google reCAPTCHA ефективно захищає веб-сайти від автоматизованих ботів, які можуть намагатися здійснювати атаки. Це зменшує ризик несанкційного проникнення у застосунок, потенійних викрадень акаунтів, спаму контенту, тощо.

Висока точність виявлення:

Завдяки передовим алгоритмам машинного навчання та аналізу поведінки користувачів, Google reCAPTCHA має високу точність виявлення ботів. Це дозволяє зменшити кількість помилкових позитивних результатів, коли реальні користувачі помилково ідентифікуються як боти.

Покращений користувацький досвід:

Google reCAPTCHA має дві версії інтеграції, V2 та V3. V2 – це явна інтеграція, де користувачу перед авторизацією необхідно пройти невеликий квест, щоб довести що це людина. Версія V3 використовує неявну інтеграцію, яка у фоновому режимі визначає чи справжня людина намагається авторизуватись.

Легка інтеграція:

Інтеграція Google reCAPTCHA у веб-застосунок є відносно простою завдяки детальній документації та готовим бібліотекам для різних платформ та фреймворків. Це дозволяє швидко впровадити захист без значних витрат часу та ресурсів.[29]

Адаптивність:

Google reCAPTCHA адаптується до змін у поведінці ботів і постійно оновлюється для забезпечення високого рівня захисту. Це означає, що розробнику не потрібно турбуватися про постійне оновлення та налаштування системи захисту, оскільки Google автоматично підтримує актуальність сервісу.[29]

Рис. 4.2 – Підключення Google ReCaptcha

ВИСНОВОК

Проаналізувавши все вище описане можна впевнено сказати, що проактивний захист повинен бути впроваджений в кожному веб застосунку без виключень. Зазвичай у тих випадках, коли розробник думає що на веб додаток не може бути сконструйована атака, чи в атаці немає змісту, і відбуваються замахи зловмисників. Веб індустрія надзвичайно стрімко розвивається, а з нею відповідно кількість та якість кібератак на різноманітні ресурси.

Зараз неможливо уявити створення веб застосунку без використання бібліотек чи фреймворків, що є досить доброю тенденцією. Та не слід покладатися повністю на таке рішення. Варто обирати тільки перевірені бібліотеки, котрі мають регулярне оновлення і аудит потенційно вразливих місць. Серед таких сторонніх рішень варто виділити декілька, які користуються великим попитом серед розробників: Angular, React, Vue.js, Svetle та Ember.js

Angular пропонує найкращий вбудований захист від XSS атак завдяки DomSanitizer, проте є складним для розробки, тому недосвідчений розробник може не знати про слабкі в його застосунку

React в свою чергу використовує віртуальний DOM і має так званий Strict Mode котрий допомагає ідентифікувати потенційні загрози. Також він є простішим для розробки для більшості, проте більш залежний від зовнішніх бібліотек, котрі власне і можуть містити потенційні слабкі місця.

Vue.js має слабкі вбудовані засоби безпеки, що означає що вся відповідальність лягає на розробника. Також є сильно залежним від зовнішніх бібліотек. Це водночас дає гнучкість для розробки і додає розробнику більше відповідальності за безпеку.

Вибір фреймворка для розробки веб застосунку є виключно справою зручності для розробника, проте не варто покладатись тільки на вбудовані механізми захисту. Хоч вони і оновлюються і стають безпечнішими, атаки прогресують паралельно з ними.

При проектуванні будь якого клієнт-серверного застосунку не варто опиратись тільки на проактивний захист на фронтенді. Велика частина з усіх атак робляться безпосередньо на сервер в обхід клієнтської сторони. Це унеможливлює фронтенд хоч якимось чином вплинути чи пом’якшити атаку. Прикладом таких атак є DDoS атаки – коли зловмисники намагаються вивести з ладу сервер перевантаживши його великою кількістю трафіку з багатьох джерел. Тому при розробці застосунку слід комплексно підходити до питань безпеки і впроваджувати сильну безпеку на кожній із сторін.

ДОДАТКИ

Посилання на репозиторій з вихідним кодом: https://github.com/Andriivyn/VoteApp

Посилання на веб-застосунок: https://vote-app-2b5ab.web.app/home

ДЖЕРЕЛА

1. URL: https://cybersecurityventures.com (дата звернення: 06.05.2024)
2. Cross Site Scripting (XSS). URL: https://owasp.org/www-community/attacks/xss/ (date of access: 07.05.2024)
3. What is cross-site scripting (XSS)? URL: https://portswigger.net/web-security/cross-site-scripting (date of access: 07.05.2024)
4. Cross Site Request Forgery (CSRF). URL: https://owasp.org/www-community/attacks/csrf (date of access: 06.05.2024)
5. What is CSRF? URL: https://portswigger.net/web-security/csrf
6. What are injection attack types? URL: https://www.contrastsecurity.com/glossary/injection-attack-types (date of access: 05.05.2024)
7. SQL injection. URL: https://portswigger.net/web-security/sql-injection
8. SQL Injection. URL: https://owasp.org/www-community/attacks/SQL_Injection (date of access: 06.05.2024)
9. Halfond W.G.J., Viegas J., Orso A. A Classification of SQL Injection Attacks and Countermeasures. URL: https://sites.cc.gatech.edu/home/orso/papers/halfond.viegas.orso.ISSSE06.pdf
10. Mitigating Cross-Site Request Forgery (CSRF) Attacks Using ReinforcementLearning and Predictive Analytics. 2023. (Applied Research in Artificial Intelligence and Cloud Computing. URL: https://researchberg.com/index.php/araic/article/view/189/169
11. What is Broken Access Control Vulnerability And How to Prevent it. URL: https://www.authgear.com/post/what-is-broken-access-control-vulnerability-and-how-to-prevent-it (date of access: 07.05.2024)
12. Front-end security best practices. URL: https://griddynamics.medium.com/front-end-security-best-practices-8c47d23caf62 (date of access: 06.05.2024)
13. Summary of Findings. URL: https://www.verizon.com/business/resources/reports/dbir/2024/summary-of-findings/ (дата звернення: 07.05.2024)

14. Кібербезпека вінформаційномусуспільстві. Інформаційно-аналітичний дайджест. № 9. 2023. URL: https://ippi.org.ua/sites/default/files/2023-9.pdf
15. Hassan1 M. M., AliM. A., Bhuiyan T., Sharif M. H. Quantitative Assessment on Broken Access Control Vulnerability in Web Applications. International Conference on Cyber Security and Computer Science (ICONCS'18), Oct 18-20, 2018 Safranbolu, Turkey (date of access: 02.05.2024)
16. Content Security Policy (CSP). URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP (date of access: 06.05.2024)Cyber Security Statistics The Ultimate List Of Stats Data, & Trends For 2023. URL: https://purplesec.us/resources/cyber-security-statistics/ (date of access: 07.05.2024)
17. OWASP Top Ten. URL: https://owasp.org/www-project-top-ten/
18. Перелік кібератак. URL: https://uk.wikipedia.org/wiki/Перелік_кібератак
19. Tiaga A., Zaeem R.N., Barber K.S. Proactive Identity Knowledge and Mitigation System. 2021. URL: https://identity.utexas.edu/sites/default/files/2021-12/proactive-identity-knowledge-and-mitigation-system_0.pdf (date of access: 06.05.2024)
20. The Role of Front-End Development in Cybersecurity: Protecting User Data. URL: https://moldstud.com/articles/p-the-role-of-front-end-development-in-cybersecurity-protecting-user-data (date of access: 06.05.2024)
21. Front-end security best practices. URL: https://griddynamics.medium.com/front-end-security-best-practices-8c47d23caf62 (date of access: 06.05.2024)
22. Обізнаність зкібербезпеки (Cybersecurity Awareness). URL: https://eska.global/service/cybersecurity-awareness (дата звернення: 07.05.2024)
23. Cybersecurity in Critical Infrastructure Sectors: A Proactive Approach to Ensure Inevitable Laws and Regulations Are Effective. 2016. 14 Colo. Tech. L.J. 345 .
24. Cross-Site Request Forgery. URL: https://knowledge-base.secureflag.com/vulnerabilities/cross_site_request_forgery/cross_site_request_forgery_vulnerability.html (date of access: 07.05.2024)

25. Які країни проводять найбільше політизованих кібератак у світі: Україна увійшла до п'ятірки. URL: https://tech.liga.net/ua/other/article/yaki-krainy-provodiat-naibilshe-polityzovanykh-kiberatak-u-sviti-ukraina-uviishla-do-piatirky (дата звернення: 07.05.2024)
26. Angular documentation. Angular. URL: https://v17.angular.io/guide/security (дата зверненя 09.05.2024)
27. Trusted types. URL: https://w3c.github.io/trusted-types/dist/spec/ (date of access 09.05.2024)
28. Firebase Documentation. Firebase. URL: https://firebase.google.com/docs (date of access: 9.05.2024).
29. reCAPTCHA v3 | Google for Developers. Google for Developers. URL: https://developers.google.com/recaptcha/docs/v3 (date of access: 09.05.2024).