МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КІБЕРБЕЗПЕКИ

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

на тему:

«Дослідження захищеності вебресурсів, створених на базі CMS-систем»

Завідувач випускаючої кафедри **Любчак В.О.**

Керівник роботи **Лаврик Т.В.**

Студент гр. КБ-81-0 **Омельченко Є.О.**

СУМИ 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра кібербезпеки

Затверджую_____

Зав. кафедрою Любчак В.О.

“_____” _____2022 р.

ЗАВДАННЯ

до випускної роботи

Студента четвертого курсу, групи КБ-81-0 спеціальності “Кібербезпека” денної форми навчання Омельченка Євгенія Олександровича.

Тема: “ Дослідження захищеності вебресурсів, створених на базі CMS-систем ”

Затверджена наказом СумДУ

№_____ від _____2022 р.

Зміст пояснювальної записки: 1) аналітичний огляд систем керування вмістом та їх інструментів для створення безпечних вебресурсів; 2) створення вебресурсів з використанням систем керування вмістом; 3) опис методів та інструментарію для дослідження захищеності вебресурсів, створених за допомогою систем керування вмістом; 4) тестування рівня захищеності вебресурів; 5) порівняльний аналіз систем керування вмістом.

Дата видачі завдання “_____”_____2022 р.

Керівник випускної роботи _____ Лаврик Т. В.

Завдання прийняв до виконання _____ Омельченко Є.О.

РЕФЕРАТ

Записка: 53 стор., 48 рис., 1 табл., 25 джерел.

Об’єкт дослідження — системи керування вмістом.

Мета роботи — дослідити захищеність вебресурсів, створених за допомогою систем керування вмістом за такими критеріями:

• збір інформації про вебресурс;

• доступність HTTP методів;

• можливість введення кроссайтингових скриптів;

• захищеність від SQL ін’єкцій.

Методи дослідження — метод аналітичного огляду, метод моделювання, метод порівняння.

Результати — проаналізовано поширені системи керування вмістом на предмет їх можливостей для створення захищених вебресурсів; створено вебресурси з використанням систем керування вмістом та проведено аналіз їх захищеності. У процесі дослідження за допомогою спеціалізованого інструментарію протестовано кожен зі створених вебресурсів відповідно до визначених критеріїв, проаналізовано результати захищеності вебресурсів, створених на базі WordPress та Joomla.

СИСТЕМА КЕРУВАННЯ ВМІСТОМ, АНАЛІЗ ЗАХИЩЕННОСТІ, ПЛАГІН, WORDPRESS, JOOMLA, DRUPAL, ВЕБРЕСУРС, HTTP МЕТОД, SQL ІН’ЄКЦІЯ, XSS

ЗМІСТ

ВСТУП

З розвитком технологій засоби зберігання та передачі інформації у мережі стали головними факторами, які визначають загальний рівень розвитку суспільства та держави в цілому. Таким чином, можна сказати, що інформація є одним із головних ресурсів, за допомогою якого можливо здійснити будь-які дії, а безпеку інформаційного простору головно складовою безпеки будь-якої країни. З кожним роком зростає роль інформації в сучасному суспільстві, постійне вдосконалення інформаційних технологій актуалізують питання щодо інформаційної безпеки.

Все більше компаній почали створювати власні вебресурси, що спочатку слугували як візитівка, яка знаходиться у мережі, та з плином часу ці вебресурси почали використовуватися в якості основного інструмента у бізнесі, за допомогою якого можна було отримати доступ до необхідної інформації, контактувати з клієнтами, підтверджувати замовлення тощо.

Хоча в сучасному світі багато людей мають особисті вебресурси, але далеко не кожен спроможний створити вебресурс з нуля. Для таких користувачів були розроблені Content Management System (CMS) системи, які є зручними та інтуїтивно зрозумілими інструментами для створення вебресурсів, котрі не потребують спеціальних знань для їх ефективного використання. У мережі існує безліч CMS систем, кожна з яких відрізняється від іншої та має свої спеціальні інструменти.

З кожним днем різноманітних вебресурсів стає все більше, що підтверджує те, що у мережі Інтернет є перспективи для розвитку. Але, з іншого боку, це надає більше можливостей для кіберзлочинців. Тим самим, забезпечення безпеки користувачів у мережі стає сьогодні нагальною необхідністю.

Метою роботи є аналіз систем керування вмістом вебресурсів з точки зору їх можливостей для створення безпечних продуктів, а також проведення дослідження створених вебресурсів з використанням спеціалізованого інструментарію на їх захищеність від кібератак.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Аналіз можливих загроз для вебресурсів

У світі існує велика кількість вебресурсів, на які може бути здійснена кібератака. На перший погляд, може здатися, що шанс бути атакованим настільки низький мотивуючи це тим, що нікому не буде цікавий звичайний вебресурс, але завжди є шанс того, що веб-ресурс буде тренувальним майданчиком для кіберзлочинців. Тому завжди необхідно звертати увагу на безпеку інтернет-ресурсу у першу чергу, для того, щоб зберегти цілісність і конфіденційність інформації, яка знаходиться в середині веб-ресурсу. Таким чином кожен власник повинен бути ознайомлений з рекомендаціями, що до безпеки.

Як приклад того, що захист вебресурсів є необхідністю, можна привести DDoS-атаку на вебресурс Президента України, яка була нейтралізована спеціалістами з кібербезпеки та це не єдина атака, яка була направлена на державні та комерційні вебресурси. З листопада 2019 року у мережі ширився ботнет під назвою KashmirBlack, який атакував веб-ресурси, які використовували популярні системи керування вмістом, він використовував вже відомі вразливості на серверах жертви, в середньому виконуючи мільйони атак за день, тим самим нагадуючи користувачам про необхідність дотримуватися встановлених правил безпеки [1].

В системах керування вмістом (Content Management System, CMS) більшість експлойтів припадає на сторону браузера, наприклад довільний код виконується браузерами локально, що може призвести до непередбачуваних наслідків, таких, як виток інформації або викликати критичну помилку. Також додаткові ризики для користувачів CMS систем створюють спливаючі вікна, плагіни та додатки, які працюють у веб-браузерах. Велика кількість таких атак можлива через те, шо кібер-злочинці викрадають акаунт розробника та випускають оновлення, в яких доданий небезпечний код. Не слід надавати

доступ до коду доповнень кінцевим користувачам і необхідно забезпечити розробку додатків за допомогою API WCMS.

Безпека CMS бути порушена за допомогою стороннього коду на стороні клієнта. Основна проблема полягає в тому, що більшість додатків включають в себе сценарії, які викликають інтерфейси API для оновлення серверних даних систем керування вмістом, за допомогою чого кіберзлочинець може надіслати шкідливий запит. Одним із прикладів є SQL команди, несподівані формати та сценарії. Таким чином було зламане API Drupal [2].

Однак більшість людей вважає, що оскільки WordPress, Drupal та Joomla є популярними і через свою популярність вони повинні мати надійний захист. Однак через свою популярність вони є привабливими мішенями для кібер-злочинців. Як приклад можна навести систему керування вмістом під назвою WordPress, яка містить у своєму ядрі, темах та плагінах приблизно 14 000 тис. відомих вразливостей. Приблизно 55% усіх атак проводяться через встановлені плагіни, які мають слабкий захист або були покинуті своїми розробниками. Також однією з головних вразливостей є встановлення простого логіну та пароля, що надає змогу зламати вебресурс за допомогою простого підбору даних відомого як “Brute force” [10]. Одним із надійних способів захисту, який може надати перевагу, – це використання без головного підходу, який відповідає за універсальний контент (бекенд), за допомогою якого можна автоматизувати розповсюдження контенту на інші платформи, які використовують дане «тіло».

Системи керування контентом, які використовують такий підхід, надають можливість до більш жорсткого контролю за доступом до самої CMS.

Попри все описане вище, необхідно оновлювати додатки, сценарії та інструменти. Через те, що старі версії можуть містити в собі лазівки, які закриваються з виходом нової версії, тим самим зменшуючи шанси того, що ваш сайт буде зламано, але необхідно пам’ятати, що оновлення може бути випущене без відома автора і воно може містити небезпечний код. Ще одним способом надійного захисту, є паролі які відповідають усім сучасним стандартам, а саме,

пароль повинен містити велику літеру, спец-символ, маленьку літеру, цифру та бути завдовжки в вісім символів. Для збільшення надійності, розташування символів повинно бути довільним. Бувають випадки, коли людина не може придумати надійний пароль, в таких випадках більшість сайтів пропонують користувачу випадково згенерований пароль, використання якого унеможливить швидкий злам акаунту користувача та підвищить рівень безпеки даних, які зберігаються на сайті [3].

1.2 Загальна характеристика систем керування вмістом

Усі CMS системи виконують одну і ту саму функцію – швидке створення вебресурсів без спеціальних знань. Розглянемо деякі найпоширеніші CMS системи.

1.2.1 WordPress

WordPress – це система керування вмістом з відкритим кодом, яка використовує мову програмування PHP та СУБД MySQL, які підтримуються більшою частиною хостингів. Однак, поміж безкоштовного використання можна купити план, який дасть більше різноманітних можливостей. Зазвичай цю CMS систему використовують для створення простих або маленьких вебресурсів, таких, як блог. Тим паче, за допомогою вбудованих інструментів, вебресурс, який був створений у WordPress можна перетворити у будь-який інший тип ресурсу.

Основні переваги:

- Велика кількість мов;
- Простота освоєння;
- Велика кількість інструментів персоналізації;
- Використання плагінів;
- Простота оновлення;
- Відкритий програмний код.

Основні недоліки:

- Час завантаження сторінки;

- Проблеми з безпекою [4].

Для розробника сторінки WordPress мають вигляд, наведений на рис. 1.1, 1.2, 1.3.

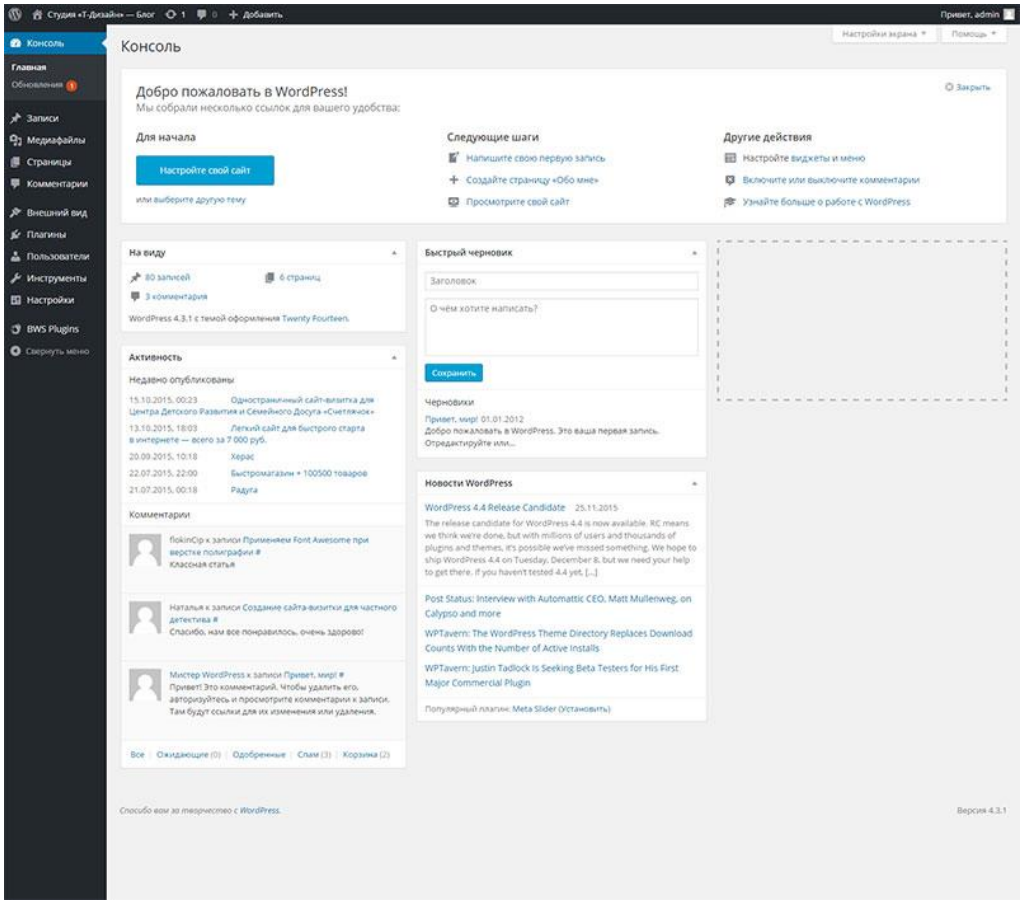

Рисунок 1.1 – Інтерфейс головної сторінки розробника.

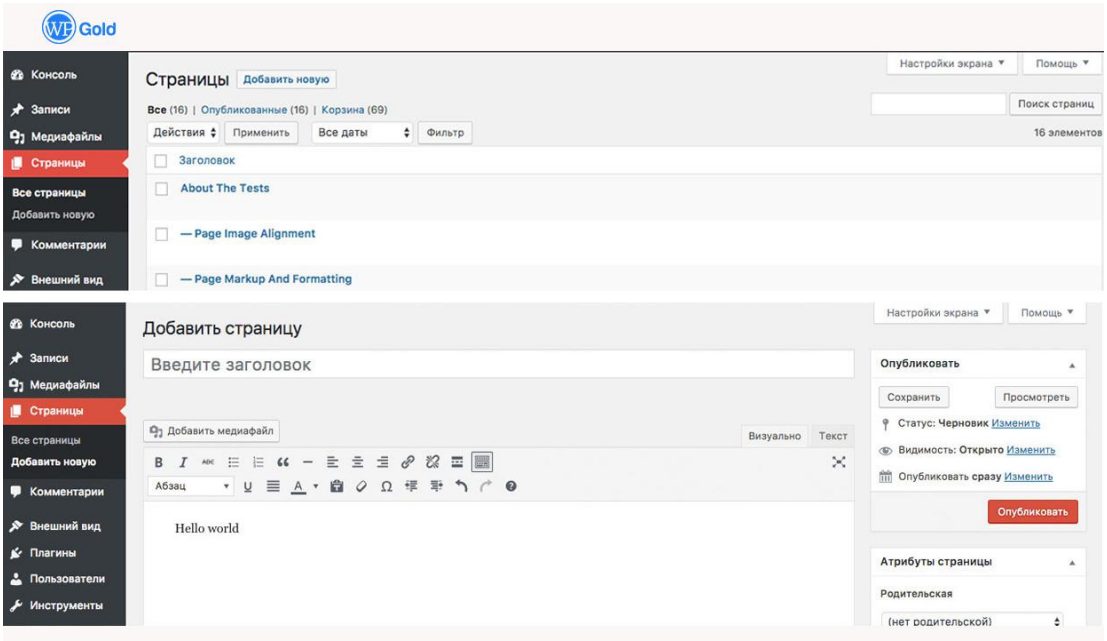

Рисунок 1.2 – Інтерфейс створення нової сторінки.

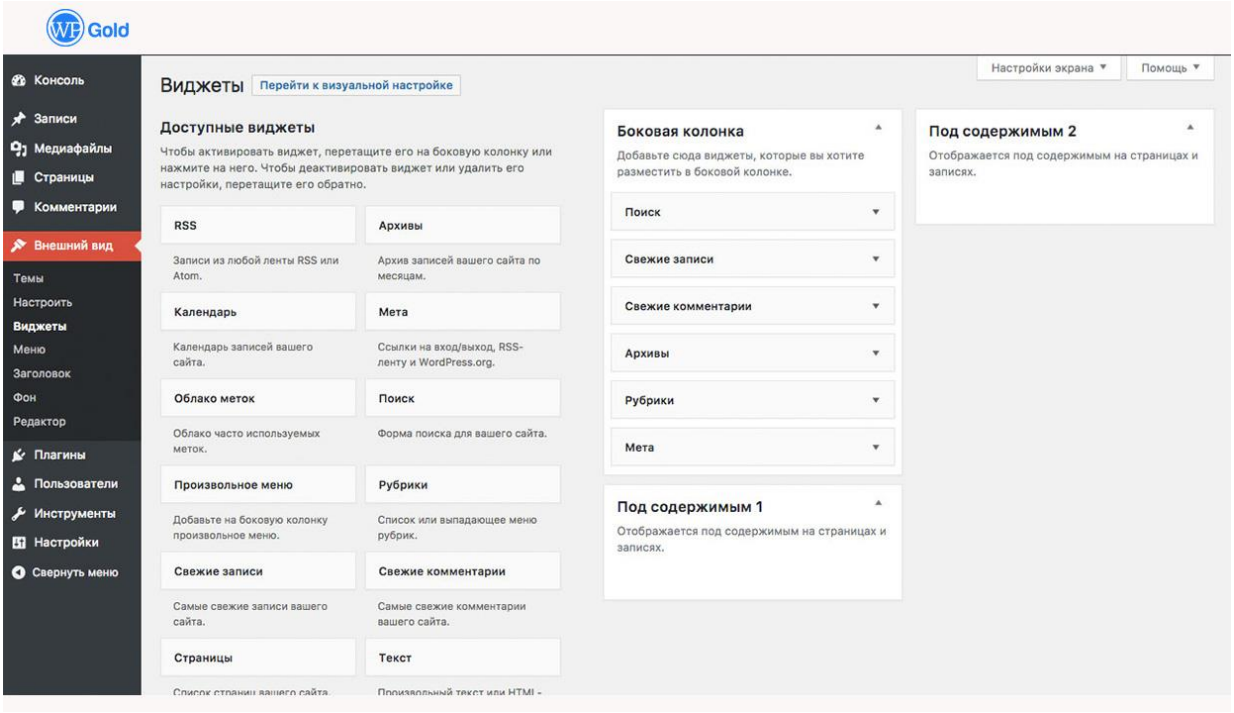

Рисунок 1.3 – Інтерфейс налаштування зовнішнього виду.

1.2.2 Drupal

Система керування вмістом під назвою Drupal була заснована на стеці LAMP, який є набором програмного забезпечення з відкритим кодом [6]. Drupal має модульну структуру, яка надає змогу змінювати зовнішній вид вебресурсу, основою якого є ядро, яке містить в собі PHP скрипти, які необхідні для коректної роботи CMS.

Основні переваги:

- Можливість роботи на пристроях з різною конфігурацією;
- Легкість налаштування, модифікації та оновлення веб-ресурсу;
- Доступність;
- Швидкість роботи;
- Підтримка великої кількості мов;
- SEO-оптимізація.

Основним недоліком є те, що ця система керування вмістом є загальноспрямованою, що не надає користувачам можливості вибрати тип вебресурсу [5].

Для розробника сторінки мають вигляд, наведений на рис. 1.4, 1.5.

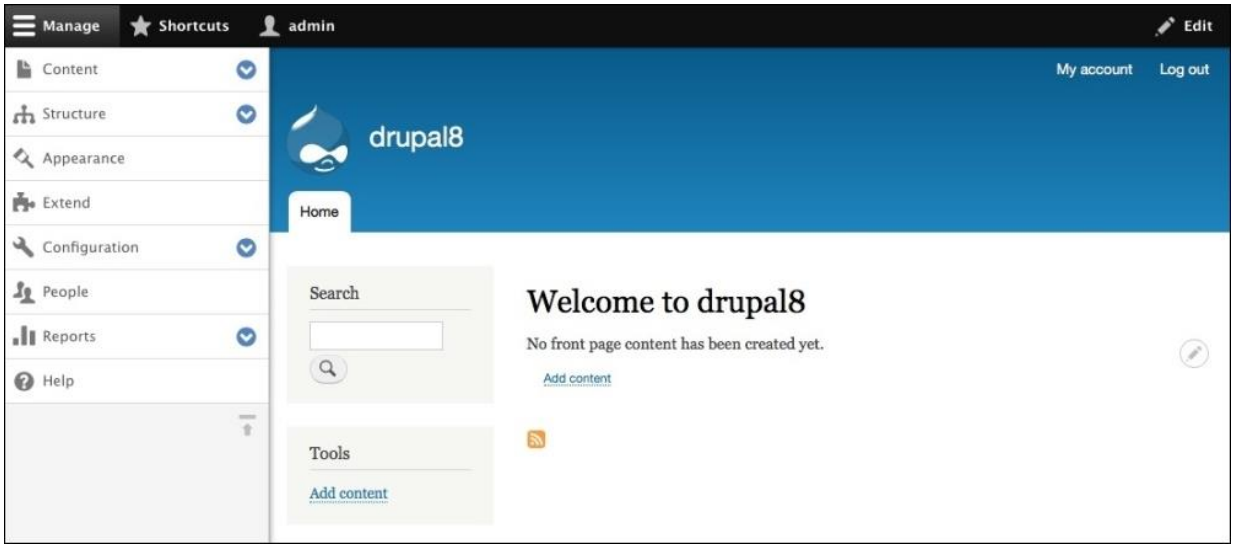

Рисунок 1.4 – Інтерфейс головної сторінки Drupal.

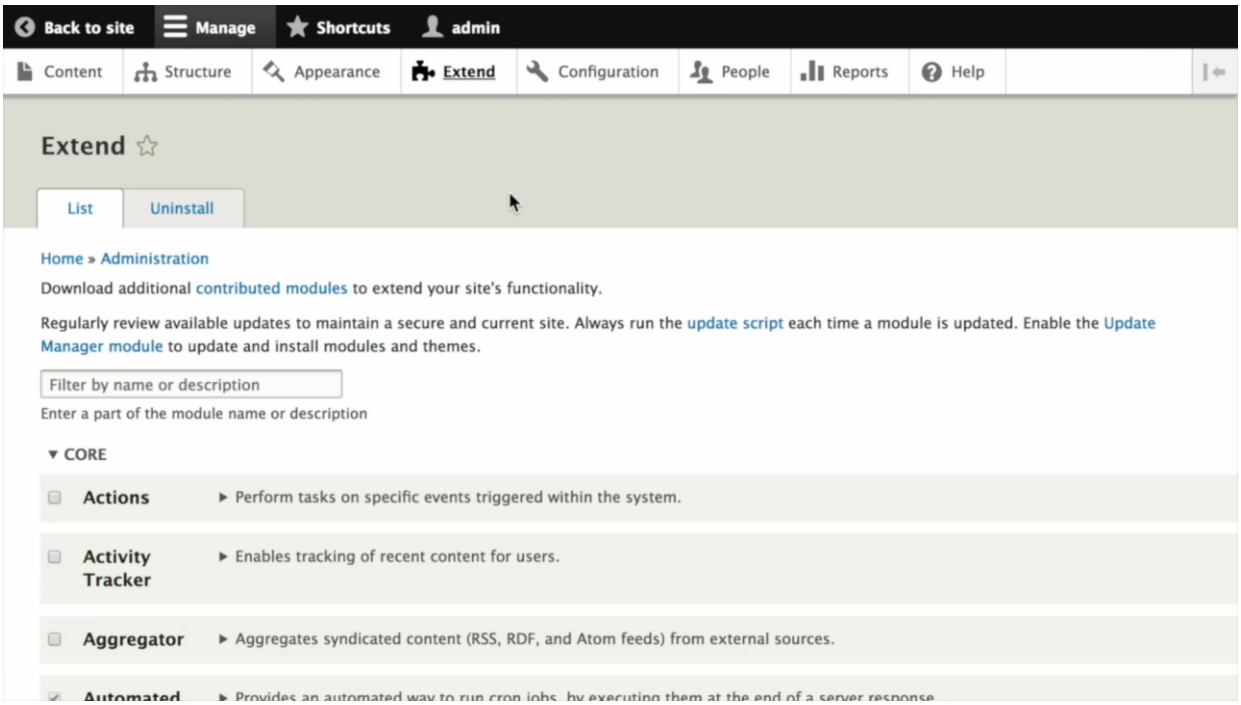

Рисунок 1.5 – Інтерфейс налаштування додатків.

Однак, незважаючи на всі переваги, що надає ця система керування контентом, вона, як й інші системи керуванням вмістом має свої недоліки. Як приклад можна навести вразливість, що була знайдена у 2014 році та отримала назву «DRUPALGEDDON». Виявлена вразливість надавала можливість відправляти спеціально створені SQL-запити напряму до бази даних, що могло призвести до несправності у роботі PHP скрипта. Також у Drupal були проблеми

із захистом від відомої вразливості Cross-Site Scripting (XSS), яка була знайдена в одній з останніх версій. Ця вразливість надавала змогу дистанційно впроваджувати шкідливі скрипти без дозволу власника або користувача [11].

1.2.3 Joomla

Joomla, як і вище описані системи керування вмістом, розповсюджується за ліцензією General Public License (GPL), що надає змогу вільно використовувати, модифікувати та розповсюджувати програмне забезпечення. Joomla була написана на мові програмування PHP та використовує такі СУБД, як MySQL, PostgreSQL або MS SQL.

При встановленні системи Joomla користувач отримує комплект необхідних інструментів для створення повноцінного вебресурсу та з плином часу цей набір буде збільшуватися завдяки іншим користувачам, які створюють свої інструменти.

Основні переваги:

- Зручне управління обліковими записами;
- Велика кількість готових шаблонів;
- Зручний інтерфейс;
- Підтримка великої кількості мов;
- Стандартизований інтерфейс;
- Функціональність [7].

Окрім переваг для Joomla виявлено декілька вразливостей, які призводили до некоректної роботи або витоку даних. Однією з таких, була вразливість системи керуванням вмістом до SQL-ін’єкцій, що викликало непередбачувані наслідки. Ця вразливість була знайдена у 2017 році. Також можна згадати про вразливість, яка надавала можливість створити акаунт з правами адміністратора, який в свою чергу надавав можливість встановити небезпечні додатки [12].

Для розробника сторінки Joomla мають вигляд, наведений на рис. 1.6, 1.7, 1.8.

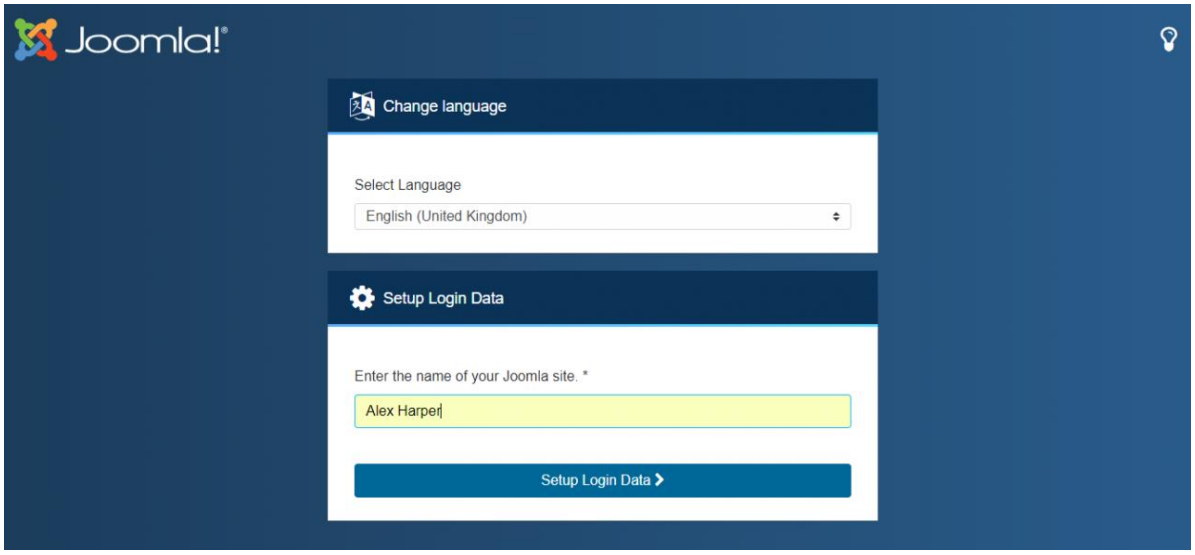

Рисунок 1.6 – Інтерфейс створення сайту.

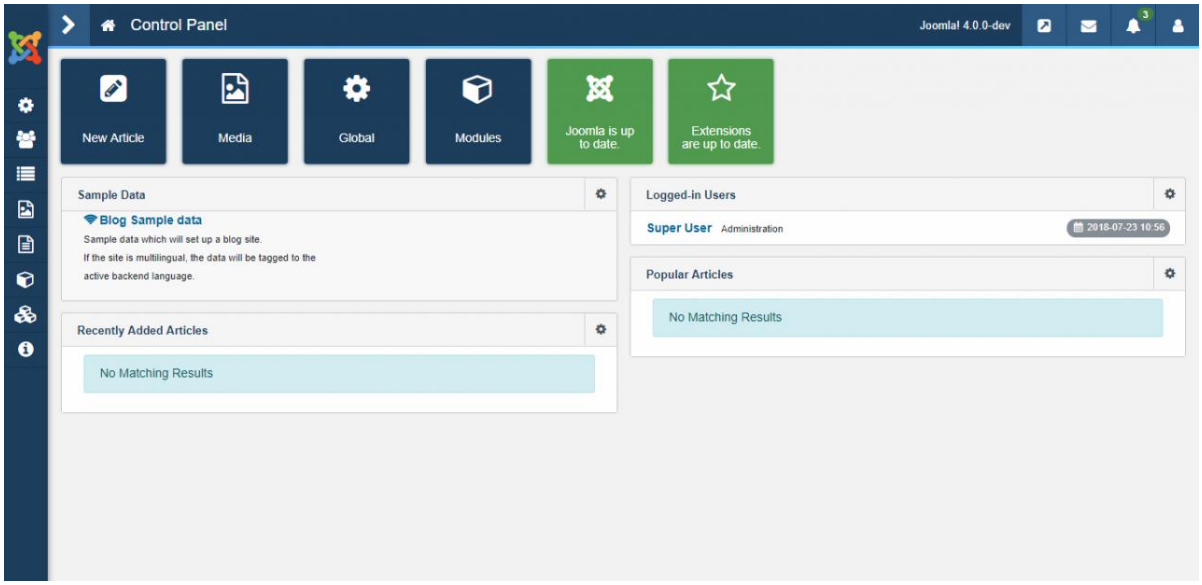

Рисунок 1.7 – Інтерфейс контрольної панелі.

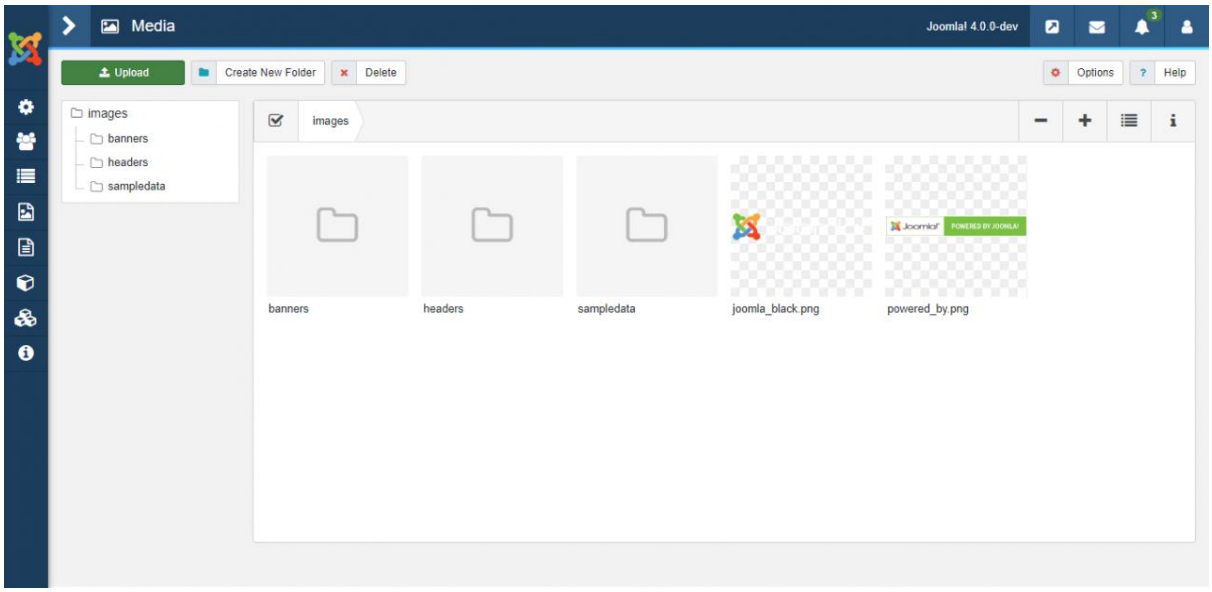

Рисунок 1.8 – Інтерфейс для вставки медіа файлів.

1.3 Постановка задачі

На основі проведеного аналізу поставлені наступні завдання:

1. Вибрати дві системи керування контентом з високим рівнем безпеки та користувацької довіри. Рівень безпеки визначається за кількістю наявних вразливостей, дані про які можна знайти в Інтернеті.

2. Дослідити можливості вибраних систем керування вмістом та створити два однакових вебресурси на їх основі.

3. Визначити етапи та інструменти для дослідження захищеності вебресурсів.

4. Провести тестування вебресурсів для аналізу їх захищеності та зробити висновки.

2 СТВОРЕННЯ ВЕБРЕСУРСІВ ЗА ДОПОМОГОЮ CMS ТА АНАЛІЗ ПЛАГІНІВ БЕЗПЕКИ

Для подальшого коректного аналізу створюємо вебресурси на базі таких CMS-систем:

– WordPress;

– Joomla.

Створення вебсайту за допомогою CMS-систем починається із завантаження цієї системи на робочий комп'ютер, на якому буде проходити створення веб-сайту, та вибору тематики створюваного вебсайту.

2.1 Створення вебресурсу за допомогою WordPress

Для початку роботи необхідно встановити та налаштувати CMS систему. Після завантаження CMS системи на локальний сервер користувач потрапляє на сторінку з описом CMS системи, після ознайомлення з якою користувач переходить на наступну сторінку, де проходить конфігурування доступу до бази даних (рис. 2.1).

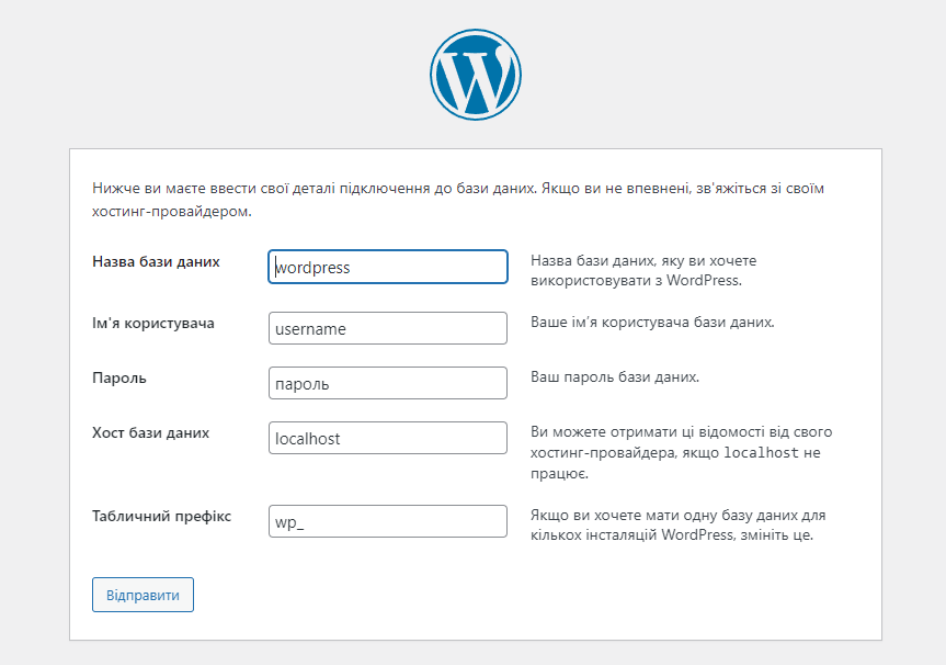

Рисунок 2.1 – Налаштування з'єднання з базою даних

Надалі необхідно вказати назву вебсайту, ім'я головного користувача, з профіля якого буде проводитися подальша робота та електрону адресу (рис. 2.2).

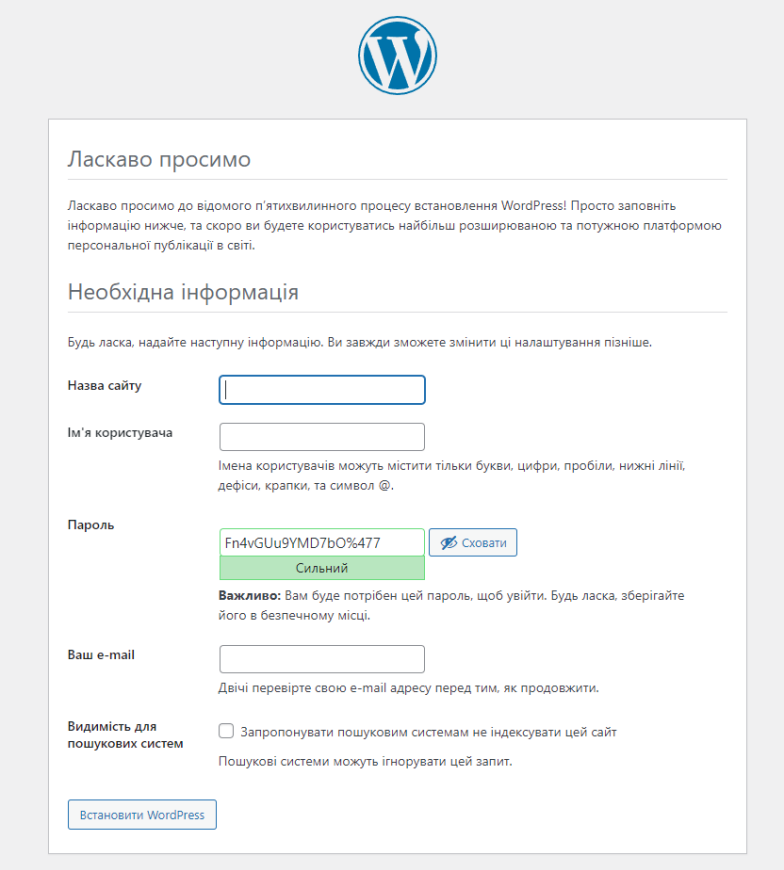

Рисунок 2.2 – Сторінка створення назви та профілю адміністратора

Відразу після створення користувача з’являється сторінка (рис. 2.3), яка повідомляє про те, що WordPress було встановлено коректно.

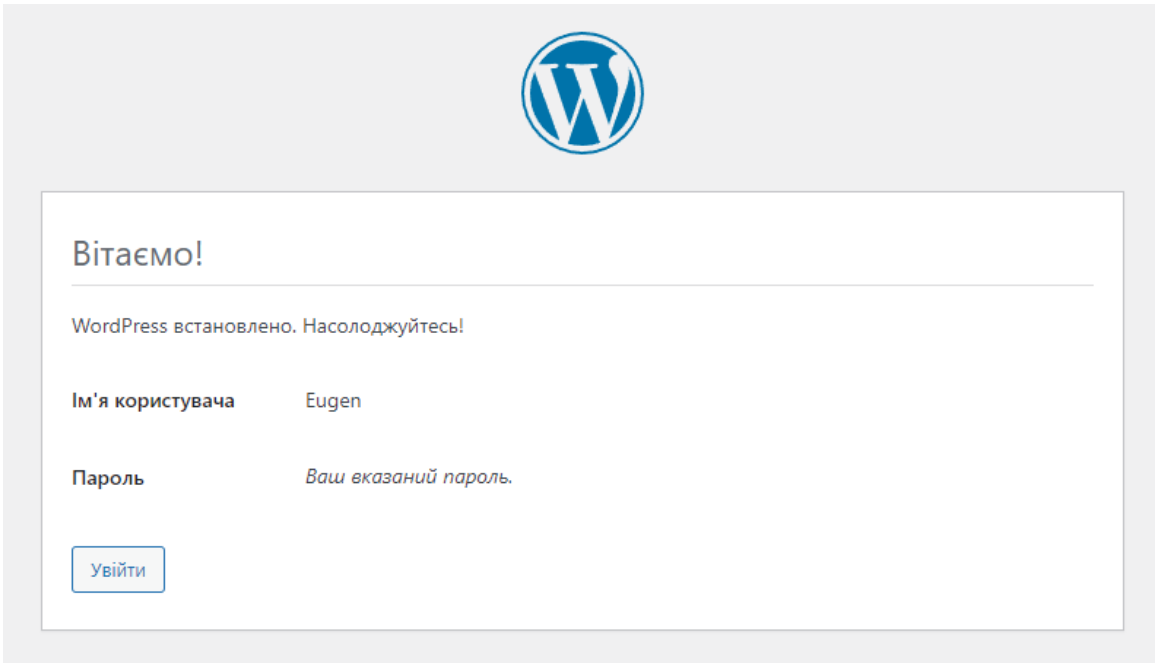

Рисунок 2.3 – Повідомлення про коректне встановлення WordPress

Наступним кроком є вхід до облікового запису користувач з правами адміністратора.

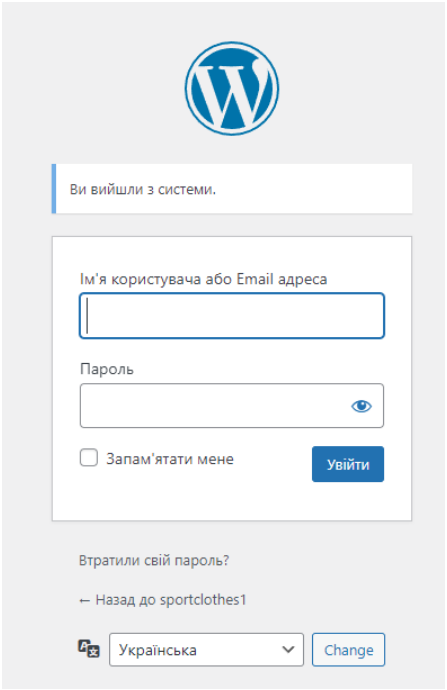

Рисунок 2.4 – Вікно входу в профіль

Одразу після введення даних для входу в профіль користувач переходить на головну сторінку (рис. 2.5), де можна буде більш детально налаштувати CMS-систему.

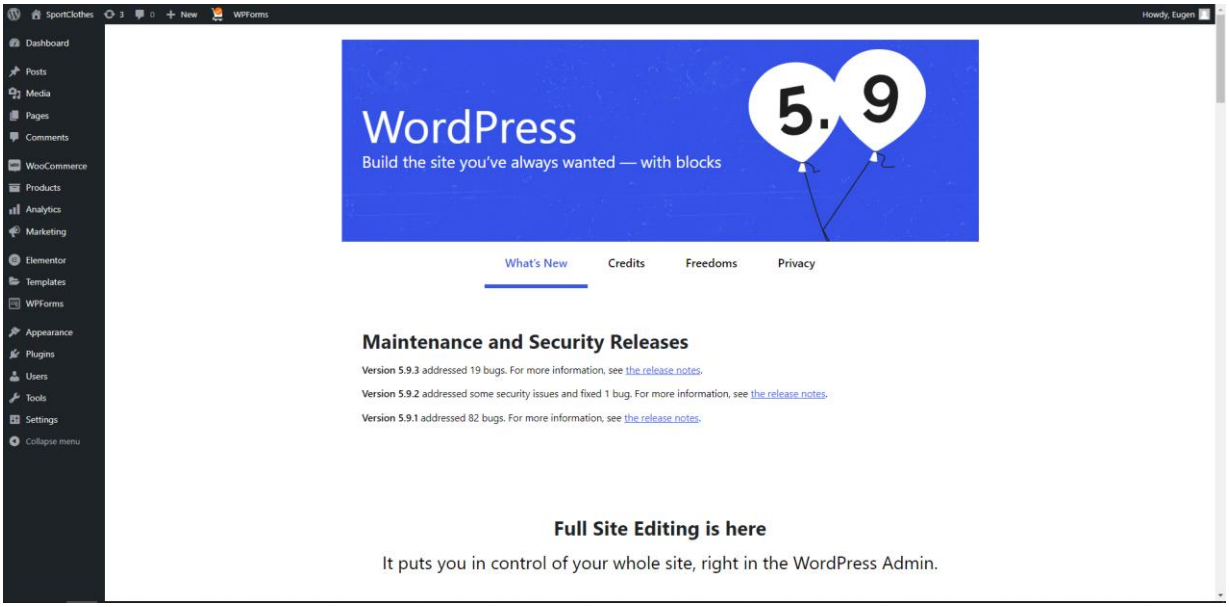

Рисунок 2.5 – Головна сторінка WordPress

Для створення сайту з мінімальними функціональними можливостями необхідно встановити тему «Astra» (рис. 2.6) та плагін Starter templates (рис. 2.7).

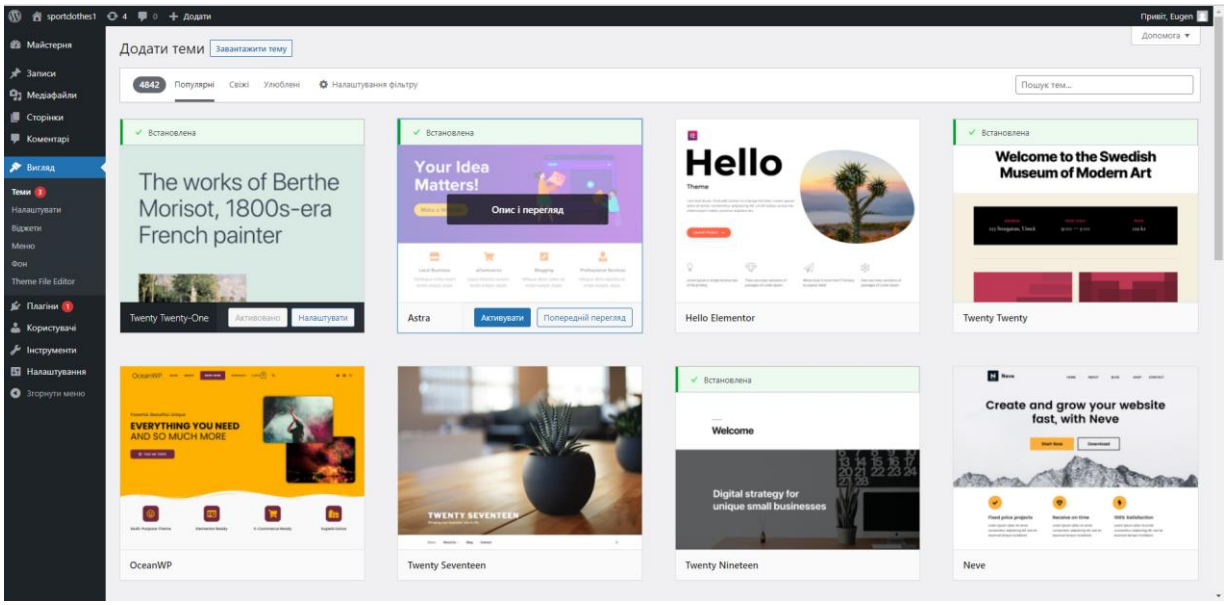

Рисунок 2.6 – Вибір теми

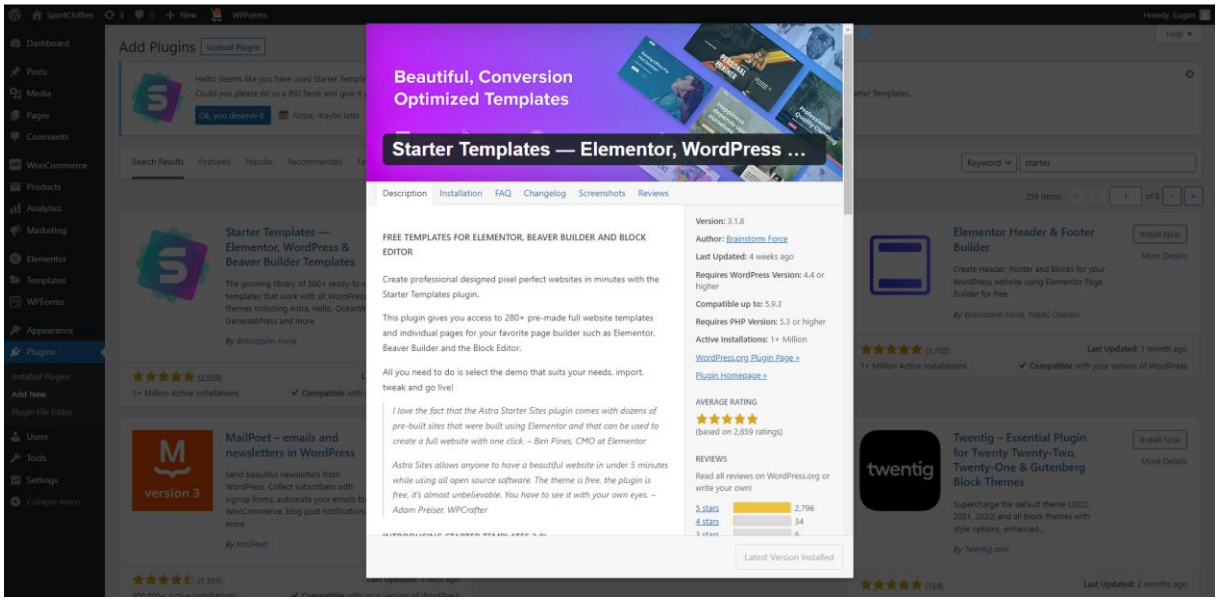

Рисунок 2.7 – Встановлення плагіну Starter templates

Після встановлення плагіну необхідно вибрати «builder» (рис. 2.8), за допомогою якого буде створено веб-сторінку.

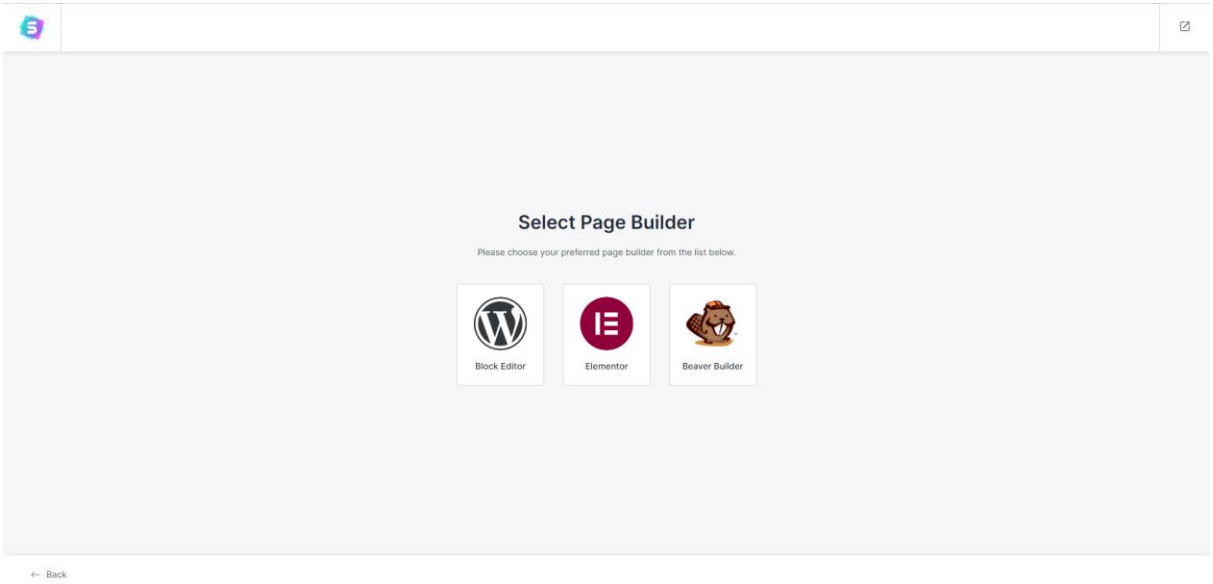

Рисунок 2.8 – Меню вибору «builder» в плагіні Starter templates

Наступним кроком користувач вибирає макет (рис. 2.9).

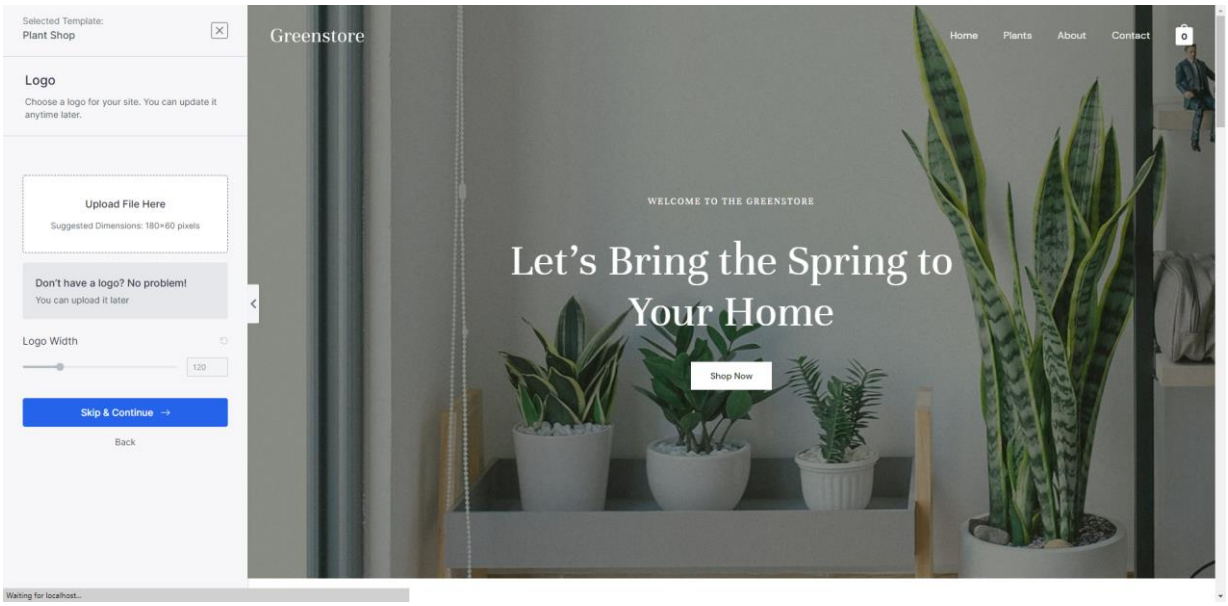

Рисунок 2.9 – Вибір шаблону

Після того, як макет обрано, буде проходити процес «будування» веб-сторінки, який відображено на рис. 2.10.

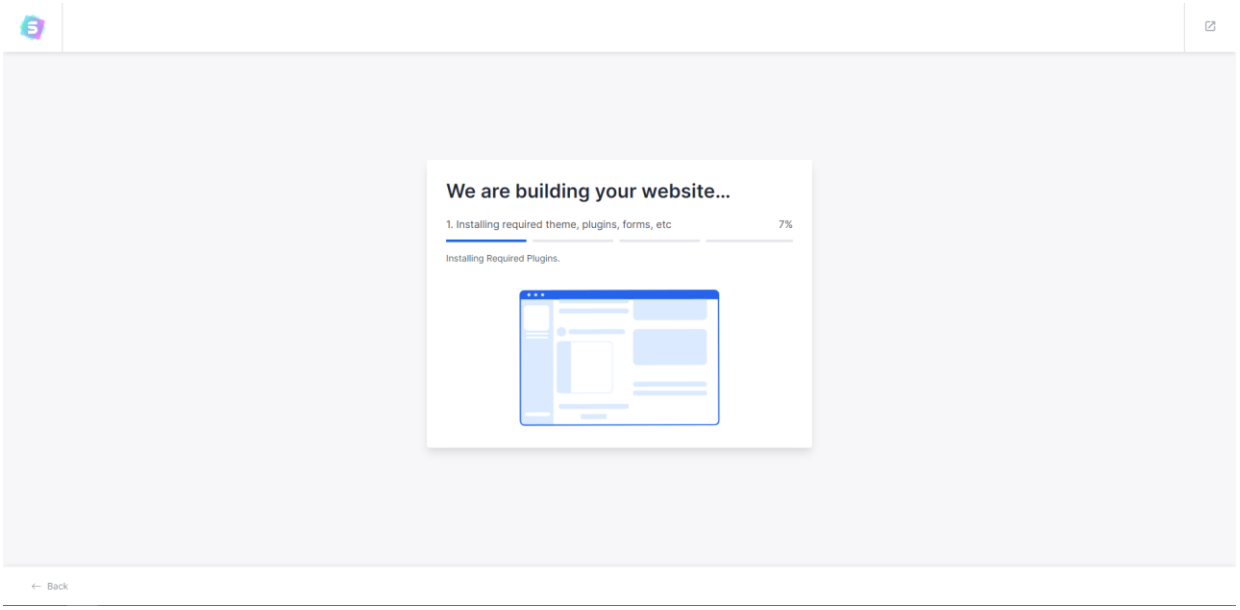

Рисунок 2.10 – Побудова веб-сторінки на базі вибраного шаблону

Для подальшої роботи необхідно змінити дизайн та наповнення веб-сторінки для того, щоб він відповідав обраній тематиці. На рис. 2.11 відображено головну сторінку веб-сайту.

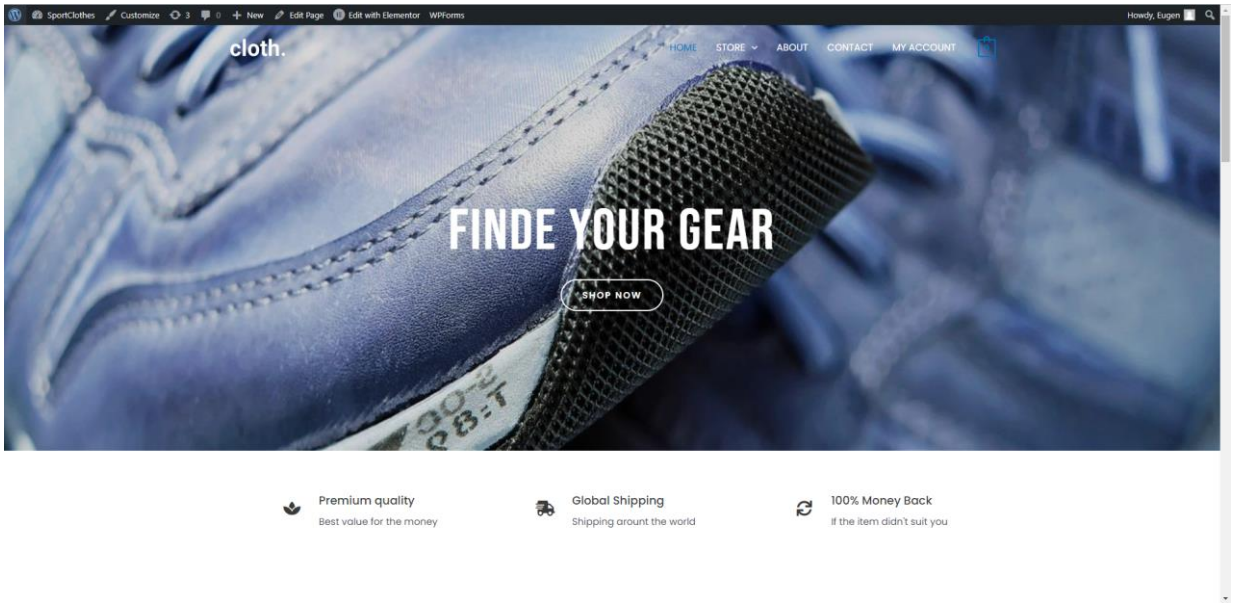

Рисунок 2.11 – Головна сторінка побудованого веб-сайту

Для покращення захищеності вебресурсу використовується плагін Jetpack. Цей плагін використовує набір інструментів, резервне копіювання в реальному часі, відновлення даних та захист від спаму (рис.2.12).

Рисунок 2.12 – Сторінка плагіну Jetpack

2.2 Створення вебресурсу за допомогою Joomla

Процес створення вебресурсу на базі CMS системи Joomla починається із завантаження цієї системи.

Процес встановлення Joomla проходить у три етапи, першим з яких є вибір назви вебресурсу та мови інтерфейсу рис. 2.13.

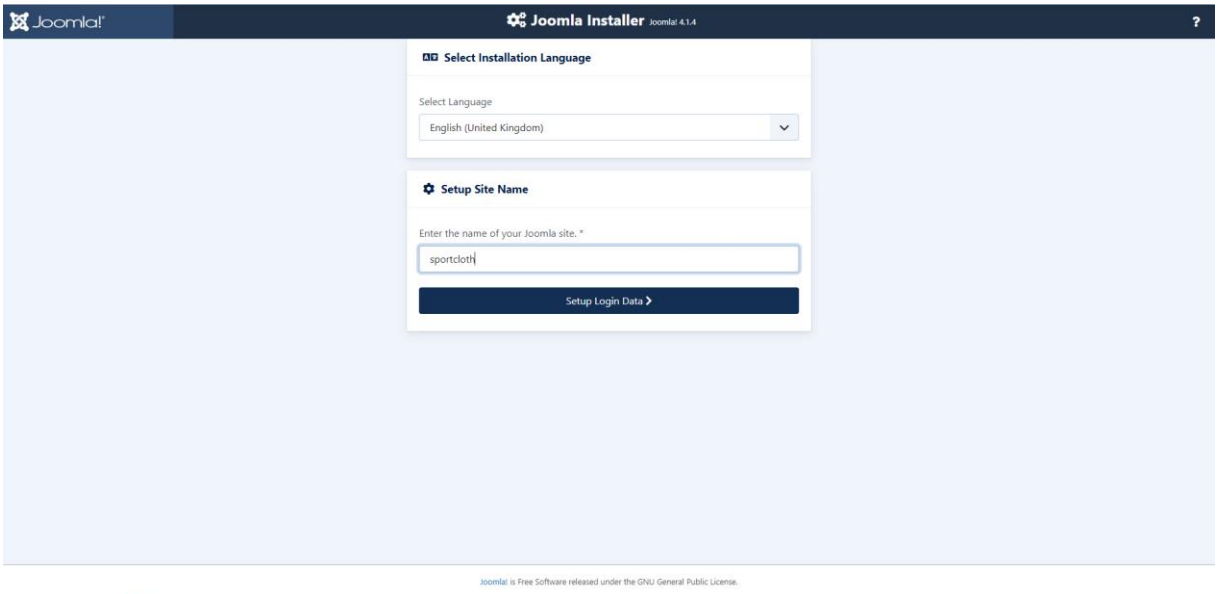

Рисунок 2.13 – Вибір назви вебресурсу

Другим етапом встановлення є створення облікового запису адміністратора, за допомогою якого буде виконуватись подальша робота (рис. 2.14).

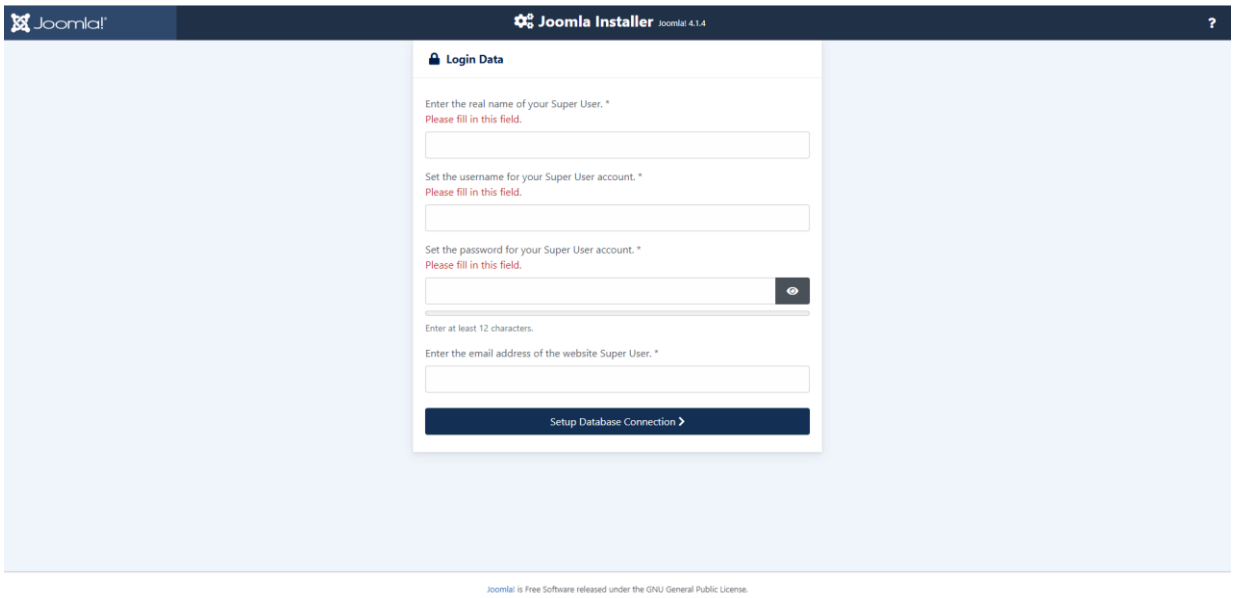

Рисунок 2.14 – Створення облікового запису адміністратора

Третім етапом встановлення є налаштування з’єднання з базою даних (рис. 2.15).

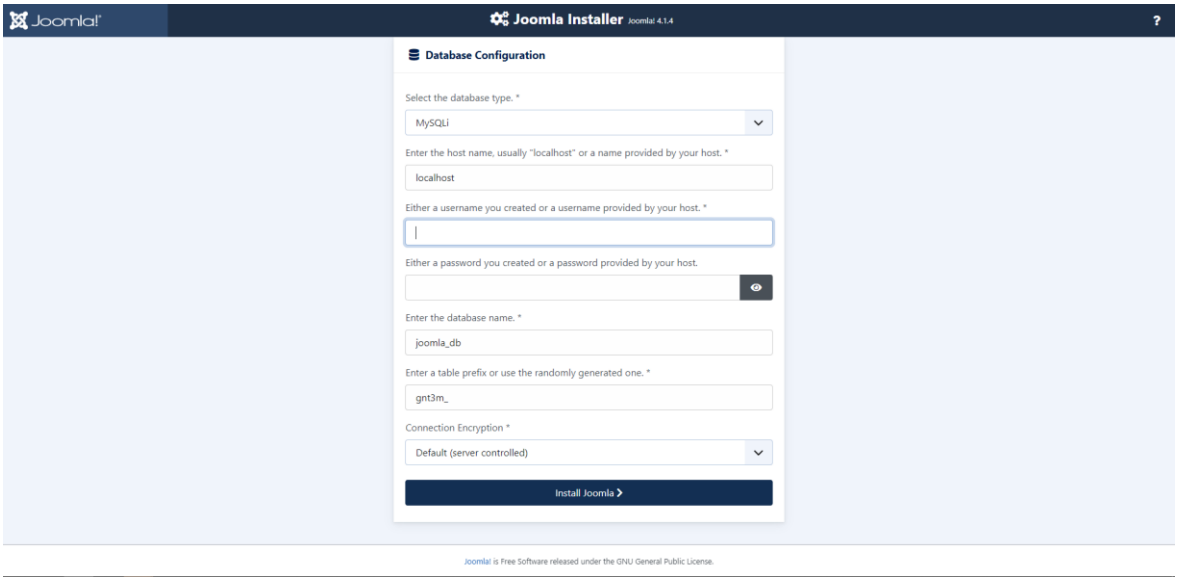

Рисунок 2.15 – Налаштування з’єднання з базою даних

Після встановлення CMS системи з’явиться вікно з повідомленням про готовність до роботи та можливістю перейти в меню адміністратора чи безпосередньо до вебресурсу (рис. 2.16).

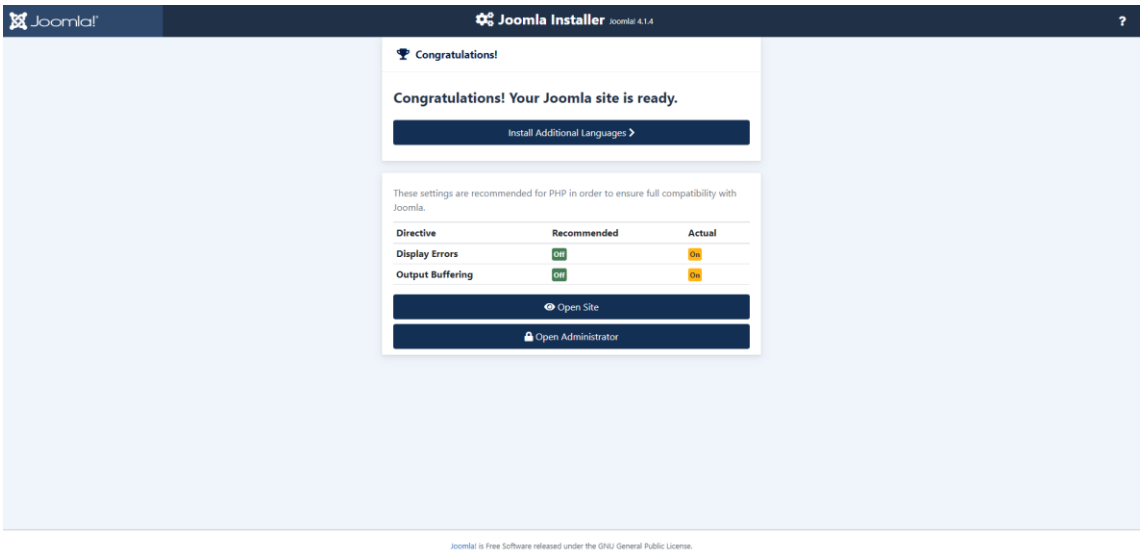

Рисунок 2.16 – Повідомлення про коректне встановлення CMS системи

Для початку безпосередньої розробки вебресурсу необхідно провести аналіз доступних та цільових плагінів, оскільки не всі плагіни та теми є безкоштовними.

Для створення вебресурсу інтернет-магазину за допомогою CMS системи Joomla знайдено та встановлено плагін HikaShop, шаблон ET_SHOES, а для поліпшення безпеки вебресурсу встановлено плагін Watchful.

2.3 Опис плагінів, використаних для розроблення вебресурсів

Starter Templates

Плагін Starter Templates надає доступ до понад 280 шаблонів та індивідуальних сторінок для різноманітних інструментів побудови сторінки. Окрім шаблонів, цей плагін надає можливість завантажити демоверсію вебресурсу. Плагін використовують більше одного мільйона людей з різними навичками та задачами через те, що він швидкодійний, легкий у налаштуванні, має піксельно-ідеальний дизайн та глибшу інтеграцію з іншими інструментами.

Основні функції, що виконує плагін:

- побудова вебресурсу на основі вибраного шаблону;
- можливість повного редагування окремих блоків сторінки;
- велика кількість різноманітних безкоштовних зображень [13].

WooCommerce

WooCommerce є однією з самих популярних платформ для інтернет-магазинів з відкритим кодом. Плагін має відкритий початковий код, що робить його гнучким та безкоштовним за рахунок великої кількості розробників.

Цей плагін допомагає створювати онлайн-магазини та має такі функції:

- можливість розробки вітрин на основі вибраної тематики;
- можливість виставлення на продаж товарів, завантаження та сортування інформації про товар;
- різноманітні способи оплати;
- налаштування процесу купівлі [14].

Jetpack

Jetpack забезпечує захист вебресурсу за допомогою комплексного набору простих засобів захисту. Плагін включає в себе можливість створення резервної копії в реальному часі та зручне відновлення, пошук шкідливого програмного забезпечення та захист від спаму. Окрім зазначених функцій, також може полегшити перехід вебресурсу на новий хостинг, створити дублікат сайту для тестування, створення резервних копій баз даних та можливість відкату до старої версії вебресурсу, має можливість вести облік внесених змін та тих, хто здійснив ці зміни [15].

HikaShop

HikaShop — це плагін, який допомагає створювати інтернет-магазини на всіх версіях Joomla.

HikaShop також має широкий спектр маркетингових інструментів, а також потужні інструменти для складання статистики, яка відображається на інформаційній панелі, щоб полегшити користувачу користування своїм інтернет-

магазином. Також цей додаток надає розширене керування податками, зонами, мовами та валютами для продажів, а також розширені можливості налаштування вашого інтернет-магазину [16].

Watchful

Watchful — це набір інструментів, який заощаджує багато часу на керування вебресурсами. Watchful дозволяє автоматизувати такі завдання, як створення резервних копій веб-сайтів, сканування на наявність ознак вторгнення та масове оновлення плагінів. Також дозволяє сканувати вебресурс на предмет використання найкращих методів безпеки, виконувати глибоке сканування, щоб знайти потенційні загрози безпеці, і створювати звіти вебресурсів для своїх клієнтів.

Основний функціонал плагіну:

- налаштування інформаційної панелі, щоб отримати огляд всього свого агентства;
- створення звіту про технічне обслуговування своїх клієнтів;
- можливість автоматичного резервного копіювання щодня, щотижня або щомісяця;
- необмежена кількість резервних копій;
- автоматичне виявлення застарілих резервних копій і помилок конфігурації резервних копій;
- постійне відстежування вебресурсів за допомогою щоденного сканування виявлення вторгнень;
- оновлення комерційних плагінів, використовуючи базу даних централізованих ключів або ліцензії оновлення [17].

3 МЕТОДИКА ТА РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1 Методи та засоби для аналізу захищеності вебресурсів

Аналіз захищеності вебресурсів може бути реалізований двома методами: ручним та автоматизованим. Ручне тестування представляє собою таке тестування, де безпосередню участь бере людина, тобто людина вибирає, які параметри тестувати та яким чином. Автоматизоване тестування представляє собою тестування з використанням спеціалізованих утиліт. Розглянемо деякі з них.

Nikto2

Nikto2 це сканер з відкритим кодом. Сканер виконує комплексне сканування веб-серверів, містить в собі список, який складається з більш ніж 6700 небезпечних файлів або програм, перевіряє 1200 серверів на застаріле програмне забезпечення та 270 серверів на специфічні вразливості, котрі притаманні лише цим серверам. Окрім цього, Nikto2 також перевіряє конфігураційні файли, файли з назвою «index», опції протоколу HTTP та може ідентифікувати встановлене програмне забезпечення та веб-сервер. Сканер дозволяє сканувати встановлені плагіни та перевіряти їх версію. Ця утиліта не є засобом для пасивного тестування, вона тестує вебсервер у швидкісному режимі [18].

OWASP ZAP

OWASP ZAP (Zed Attack Proxy) може проводити активне і пасивне сканування вразливостей. Пасивний тест це такий тест, який перевіряє вебсайт на наявність відомих вразливостей та ніяким чином не впливає на дані вебсайту, що є більш безпечним способом сканування. Активне сканування може використовувати різноманітні методи проникнення, воно може змінювати дані та вставляти небезпечні скрипти. Тому активним видом сканування краще користуватися з метою перевірки вебсайту, який запущений на локальному хості. OWASP ZAP може генерувати тестові рядки різними методами, що, в свою чергу, доволі корисно при первісному виявлені загрози. Ще однією особливістю

цього інструменту є те, що в ньому є можливість генерувати запити безпосередньо при виявлені загрози, відповідь на ці запити відображається у тому ж робочому вікні, що значно покращує швидкість прийняття рішення щодо певної загрози [19].

Sqlmap

Sqlmap використовується для тестування на проникнення та має відкритий початковий код, що за потребою дозволяє модифікувати його під нетривіальні задачі. Цей інструмент автоматизує процес виявлення та експлуатації вразливостей, також містить в собі багато різноманітних функцій від збору відбитків баз даних за отриманою від них інформації до можливості отримати доступ до файлової системи. Підтримує майже усі системи керуванням базами даних (СУБД), може напряму підключатися до баз даних, автоматично розпізнає формати зашифрованих даних, наявна можливість створення дампу бази даних та інше [20].

Nmap

Nmap це безкоштовна утиліта з відкритим кодом, яку використовують для аудиту безпеки та пошуку мереж. Ця утиліта використовує немодифіковані IP пакети, що дозволяє визначити, які хости доступні у мережі, які сервіси ці хости пропонують, яку операційну систему використовують, який файервол або фільтр пакетів використовується на даний момент. Цей інструмент розроблений для сканування великих мереж та ефективно працює з одинокими хостами. В додаток до класичного функціоналу Nmap надає просунутий інтерфейс, гнучку передачу даних, перенаправлення та інструменти для дебагінгу, також надає можливість порівнювати результати та інструмент для генерації та аналізу пакетів [21].

VegScanner

Цей інструмент є безкоштовним сканером та тестовою платформою з відкритим кодом для тестування безпеки вебресурсів, має вбудований сканер вразливостей для швидкого тестування. VegScanner допоможе знайти та

підтвердити SQL ін’єкції, кроссайтинговий скриптинг, незахищену чутливу інформацію та інше. Окрім цього може зондувати налаштування безпеки протоколу TLS/SSL і запропонувати можливі рішення для покращення захисту сервера. Розроблено цей інструмент на мові програмування Java. Він має графічний інтерфейс та сумісний з усіма відомими операційними системами [22].

WPScan

WPScan це сканер вразливостей, який має закритий початковий код і розроблений для сканування вебресурсів, створених на базі Wordpress. Цей інструмент має свою базу даних вразливостей, які присутні в WordPress. Сканер працює за принципом «чорної» скриньки і під час сканування він імітує хакерську атаку [23].

WhatWeb

WhatWeb створений для того, щоб ідентифікувати вебресурси. Метою цього інструмент є сканування вебресурсів та ідентифікація технологій, які були використані при створенні ресурсу, а також CMS систему, якщо така використовувалася. WhatWeb має більше 1700 функціонуючих плагінів, кожен з яких виконує свою задачу. Цей засіб передбачає можливість регулювання рівня агресивності, який використовується у випадках коли з першого разу не вдалося отримати усю необхідну інформацію. Більшість плагінів, що використовуються, є фундаментальними та у своїй роботі враховують велику кількість різноманітних даних. Деякі вебресурси видаляють тег <meta>, за яким можна визначити використану CMS систему, тому WhatWeb при тестуванні використовує 15 різноманітних тестів [24].

3.2 Організація дослідження

Тестування безпеки це метод оцінки безпеки системи чи мережі, що перевіряє та підтверджує ефективність системи контролю безпеки ресурсів. Тестування вебресурсів зосереджене на оцінці безпеки вебресурсів. Процес тестування включає активний аналіз вебресурсу на вразливості. Інформація про будь-яку вразливість, що виявлена під час тестування, доводиться до відома

власника вебресурсу, разом з поясненням того, як саме вразливість може впливати на безпеку та шляхи вирішення цієї проблеми.

Для повноцінного розуміння необхідності тестування потрібно уточнити, що таке вразливість, небезпека та тест. Вразливість це недолік дизайну системи, який може бути використаний для проникнення до програмного коду вебресурсу. Небезпеку для вебресурсів представляє усе, що піддає систему небезпеці, чи то зовнішні атаки, чи то внутрішня недбалість розробників. Під тестом розуміють дію, яка спрямована на перевірку системи безпеки вебресурсу на вразливість та відповідність вимогам безпеки.

Для проведення дослідження захищеності створених вебресурсів як основу використаємо методологію OWASP Web Security Testing Guide (OWASP WSTG - v4.2) [25]. Методологія OWASP базується на підході «чорної скриньки». Суть підходу «чорна скринька» полягає у том, що тестувальник володіє певною мінімальною кількістю інформації про систему, що тестується.

Згідно методології OWASP Web Security Testing Guide тестування поділяють на два види: пасивне та активне.

Пасивне тестування полягає у тому, що тестувальник намагається зрозуміти логіку побудови вебресурсу та досліджує вебресурс, як користувач. Також тестувальник може використовувати різноманітні інструменти для збору інформації. Наприклад, HTTP проксі можна використати з метою огляду усіх запитів та відповідей на них. По завершенню цієї стадії тестувальник повинен знати усі точки доступу та увесь функціонал системи.

Активне тестування поділяється на 12 категорій, кожна з яких має свою мету та спосіб реалізації.

Під час активного тестування за методологією OWASP Web Security Testing Guide передбачаються такі етапи перевірки:

- збір інформації;
- конфігурація та розгорнення тесту менеджменту;
- тестування управління ідентифікацією;

- тестування аутентифікації;
- тестування авторизації;
- тестування сесій менеджменту;
- тестування перевірки введення;
- обробка помилок;
- криптографія;
- тестування бізнес-логіки;
- тестування на стороні клієнта;
- тестування API.

Хоча за методологією OWASP WSTG передбачається перевірка у 12 етапів, кожен з яких містить ще й підетапи, для аналізу захищеності вебресурсів, створених за допомогою CMS систем, обрано саме такі етапи:

- збір інформації про вебресурс;
- тестування методів HTTP;
- тестування на можливість введення сторонніх скриптів;
- тестування на SQL ін’єкції.

Розглянемо детальніше кожен з етапів.

1. Збір інформації про вебресурс.

У кожного вебресурсе є HTML заголовки, cookies та структура директорій. Усі ці частини вебресурсу допомагають ідентифікувати, що саме було використано при розробленні вебресурсу. Більшість CMS систем мають спеціальні маркери, які допомагають відрізнити їх від інших. Це в основному те, що автоматизовані інструменти здійснюють: шукають ці маркери, а потім порівнюють із базами даних.

Зазвичай ці маркери розташовуються у таких місцях:

- заголовки HTTP;
- cookies;
- вихідний HTML код;

- специфічні файли та папки;
- розширення файлів;
- повідомлення про помилки.

Основним способом ідентифікації є перевірка поля X-Powered-By в HTTP заголовках. Іншим надійним способом є перевірка Cookies.

2. Тестування методів HTTP пропонує набір методів, які зазвичай використовуються при роботі вебресурсу. Самими відомими HTTP методами є методи POST та GET, за допомогою яких надається доступ до інформації вебсервера. Але окрім них існують інші відомі методи, за допомогою яких, зловмисник може отримати доступ до інформації.

Перелік існуючих HTTP методів:

- GET;
- POST;
- HEAD;
- PUT;
- DELETE;
- CONNECT;
- OPTIONS;
- TRACE.

Незважаючи на таку велику кількість різноманітних методів, у вебресурсах зазвичай використовують методи POST та GET, які отримують дані користувача з рядка URL, що вводяться користувачем. Стандартні посилання, як і форми, визначені без ініціації методу GET. JavaScrips та AJAX запити можуть надсилати інші методи, але зазвичай цього не роблять. Через те, що інші методи використовуються не дуже часто, більшість розробників не беруть до уваги те, як вебресурс взаємодіє з цими методами.

3. Тестування на можливість введення сторонніх скриптів.

Кроссайтовий скриптинг або XSS вразливість це такий вид вразливості в інтерактивних частинах вебресурсу, коли існує можливість введення сторонніх

скриптів. XSS вразливості виникають у ситуаціях, коли у згенеровану сервером сторінку потрапляють користувацькі скрипти. Відображені XSS як один з різновидів кроссайтового скриптингу з’являються коли зловмисник вводить виконуваний код в браузер в одному HTTP запиті. Однак код, який введено не зберігається всередині вебресурсу, і при перезавантаженні сторінки все повернеться до початкового стану. Атакуючий рядок входить до створеного URL рядка або HTTP параметра, які некоректно обробляються вебресурсом. Відображені XSS це один з найбільш використовуваних видів XSS атак. Такі атаки ще відомі, як непостійні, оскільки навантаження атаки постачається та виконується за допомогою одного запиту та відповіді, їх ще можна побачити під назвою атаки першого порядку.

Коли вебресурс вразливий до такого типу атак, він пропускає неперевірені вхідні дані. Звичайним сценарієм є надання користувачу зміненого URL посилання та переконання користувача у легітимності цього посилання. Після переходу за наданим посилання небезпечний скрипт починає діяти з використанням браузера користувача.

Однією з головних частин протидії XSS скриптам є правильне кодування символів. У деяких випадках вебсервер або вебресурс не можуть фільтрувати деякі кодування, наприклад, вебресурс може відфільтрувати <script>, але не зможе відфільтрувати %3cscript%3.

4. Тестування на SQL ін’єкції.

Тестування на SQL ін’єкції перевіряє чи можливо впроваджувати дані у вебресурс так, щоб запит виконувався на стороні клієнта. Вдала експлуатація цієї вразливості дозволяє неавторизованому користувача маніпулювати даними.

Атака за допомогою SQL ін’єкцій виконується за допомогою ін’єкції команди в підготовленому запиті. В основному такі вразливості можливі через наявність помилок у написаній розробником SQL команди, яка приймає дані від користувача. Наприклад, select title, text from news where id=$id, де змінна $id,

яка являє собою дані, введені користувачем, а інша частина запиту написана розробником та не змінюється, що робить запит динамічним.

Успішна SQL ін’єкція потребує створення такого синтаксично правильного запиту, що вебресурс не помітить втручання і виконає написаний запит. Якщо вебресурс повертає повідомлення про неправильний запит, це може допомогти зловмиснику переписати запит, так, щоб його не можна було виявити.

3.3 Результати проведеного дослідження

Для експериментальної частини було проаналізовано два вебресурси, що було створено на CMS системах Joomla та WordPress. План дослідження кожного з вебресурсів передбачав аналіз ресурсу без використання захисного плагіну та з використанням відповідного плагіну. Для кожної CMS системи обрано найпопулярніші та безкоштовні плагіни безпеки.

Аналіз захищеності розроблених вебресурсів складається з чотирьох частин. По-перше, необхідно зібрати інформацію про вебресурс, на якому сервері розташована, які технології використовувалися тощо. Для реалізації цієї задачі буде використовуватися інструмент WhatWeb. Цей інструмент має три рівні агресивності, але для збору інформації щодо вебресурсу, який тестується достатньо першого рівня.

Для того, щоб почати збір інформації необхідно ввести у консоль команду whatweb [опції] <URL>. Першим було проведено збір інформації про вебресурс на базі CMS системи WordPress з деактивованим плагіном (рис. 3.1 - 3.2).

Після отримання результатів аналізу з деактивованим плагіном було проведено збір інформації про вебресурс на базі CMS системи WordPress з активованим плагіном (рис. 3.3 – 3.4).

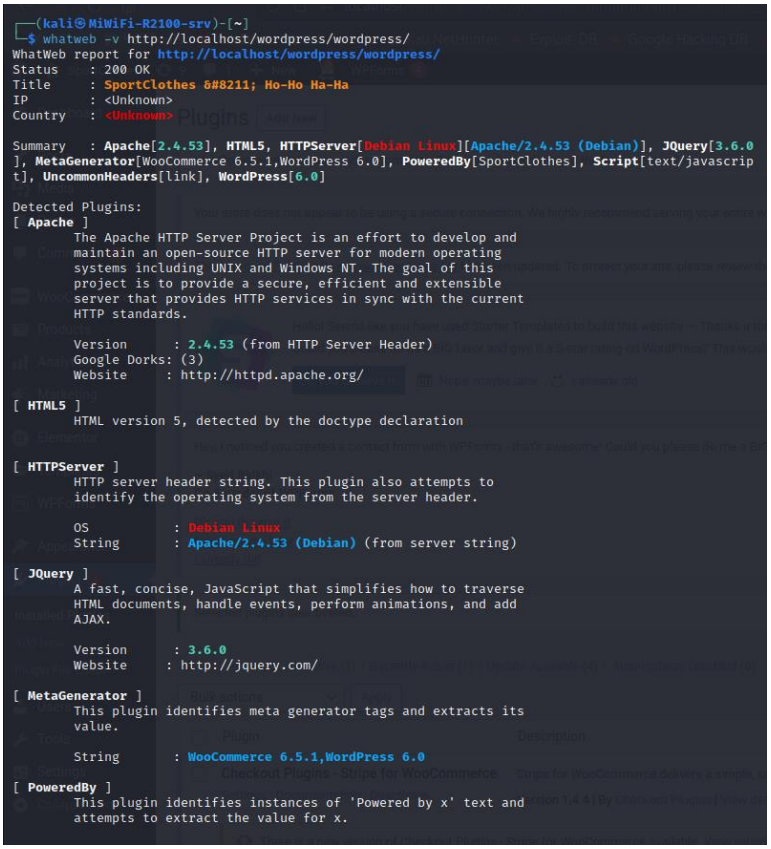

Рисунок 3.1 – Перша частина збору інформації про вебресурс без активованого захисного плагіну на базі CMS системи WordPress

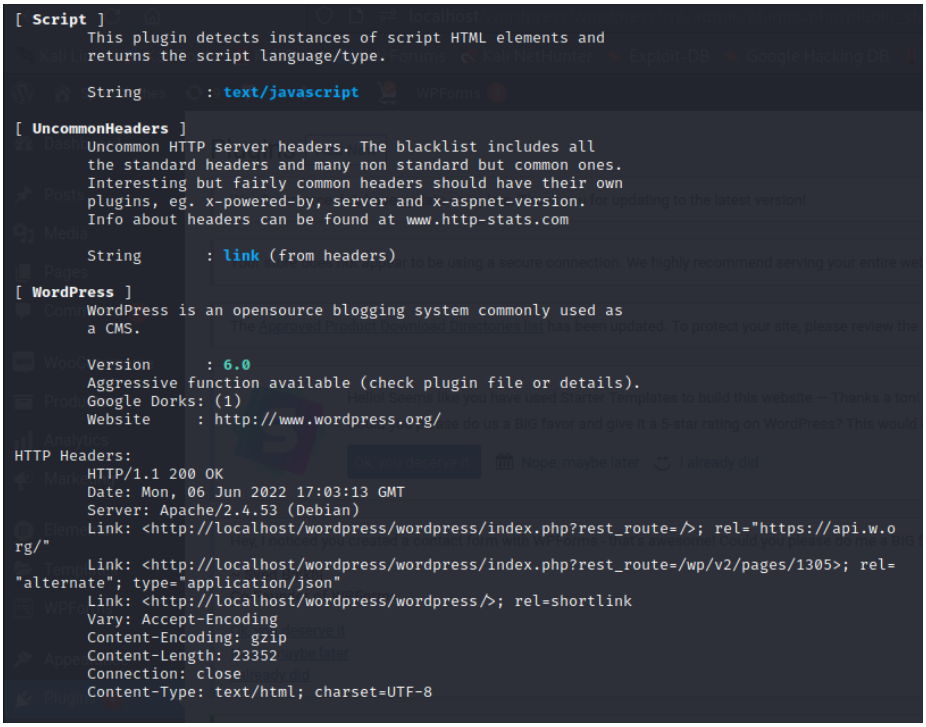

Рисунок 3.2 – Друга частина збору інформації про вебресурс без активованого захисного плагіну на базі CMS системи WordPress

Рисунок 3.3 – Перша частина збору інформації про вебресурс з активованим захисним плагіном на базі CMS системи WordPress

За результатами збору інформації про вебресурс на базі CMS системи WordPress як з використанням захисного плагіну, так і без нього, було визначено, що захисний додаток не впливає на результати зібраної інформації про вебресурс.

Після збору інформації про вебресурс, який був створений на базі CMS системи WordPress, необхідно провести збір інформації про вебресурс, який був створений на базі CMS системи Joomla, за тією ж схемою як і попередній. Результати збору інформації без захисного плагіну наведено на рис. 3.5 – 3.7.

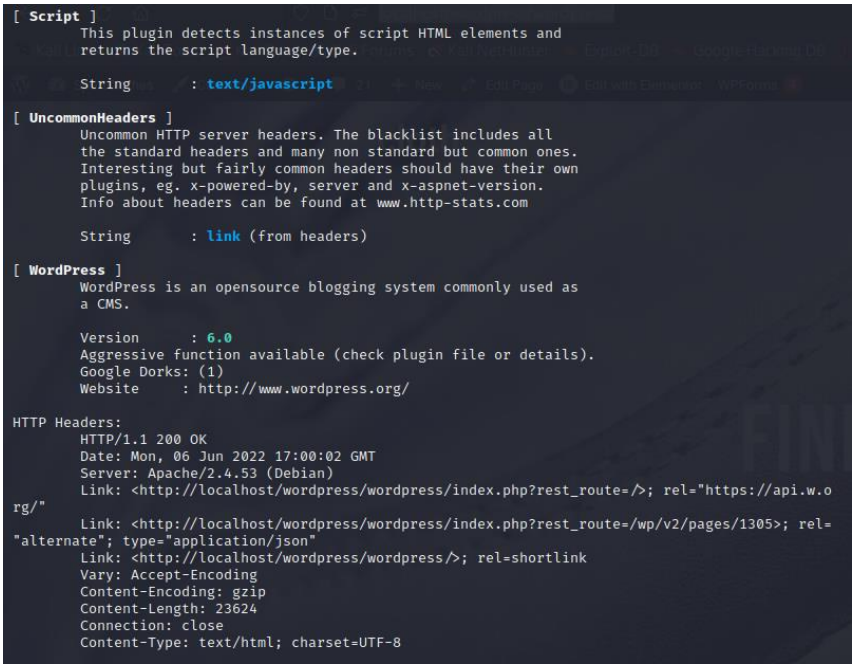

Рисунок 3.4 – Друга частина збору інформації про вебресурс з активованим захисним плагіном на базі CMS системи WordPress

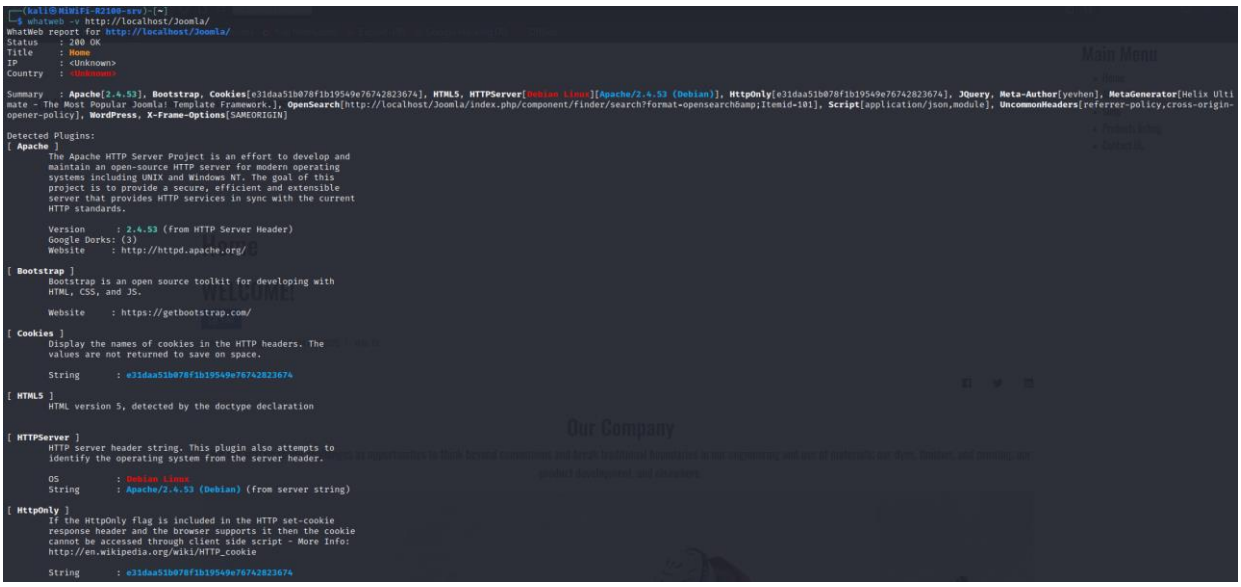

Рисунок 3.5 – Перша частина збору інформації про вебресурс без активованого захисного плагіну на базі CMS системи Joomla

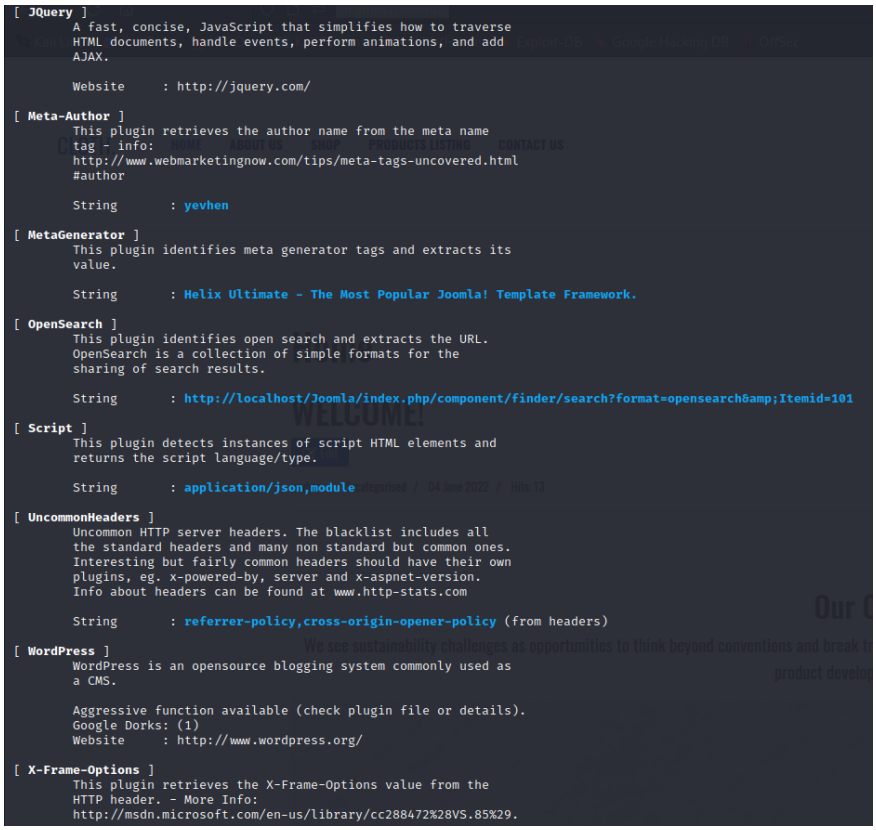

Рисунок 3.6 – Друга частина збору інформації про вебресурс без активованого захисного плагіну на базі CMS системи Joomla

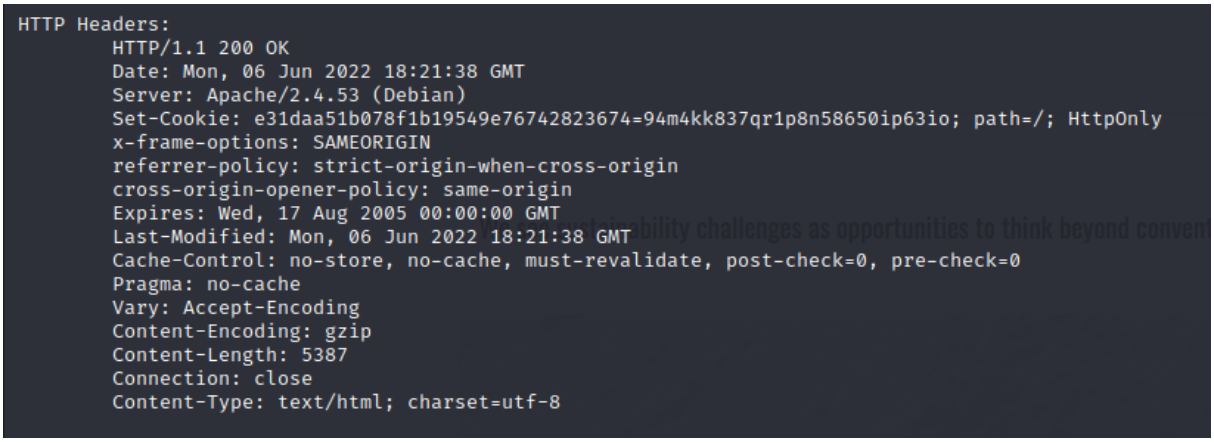

Рисунок 3.7 – Третя частина збору інформації про вебресурс без активованого захисного плагіну на базі CMS системи Joomla

Результати збору інформації з активним захисним плагіном зображено на рис. 3.8 – 3.10.

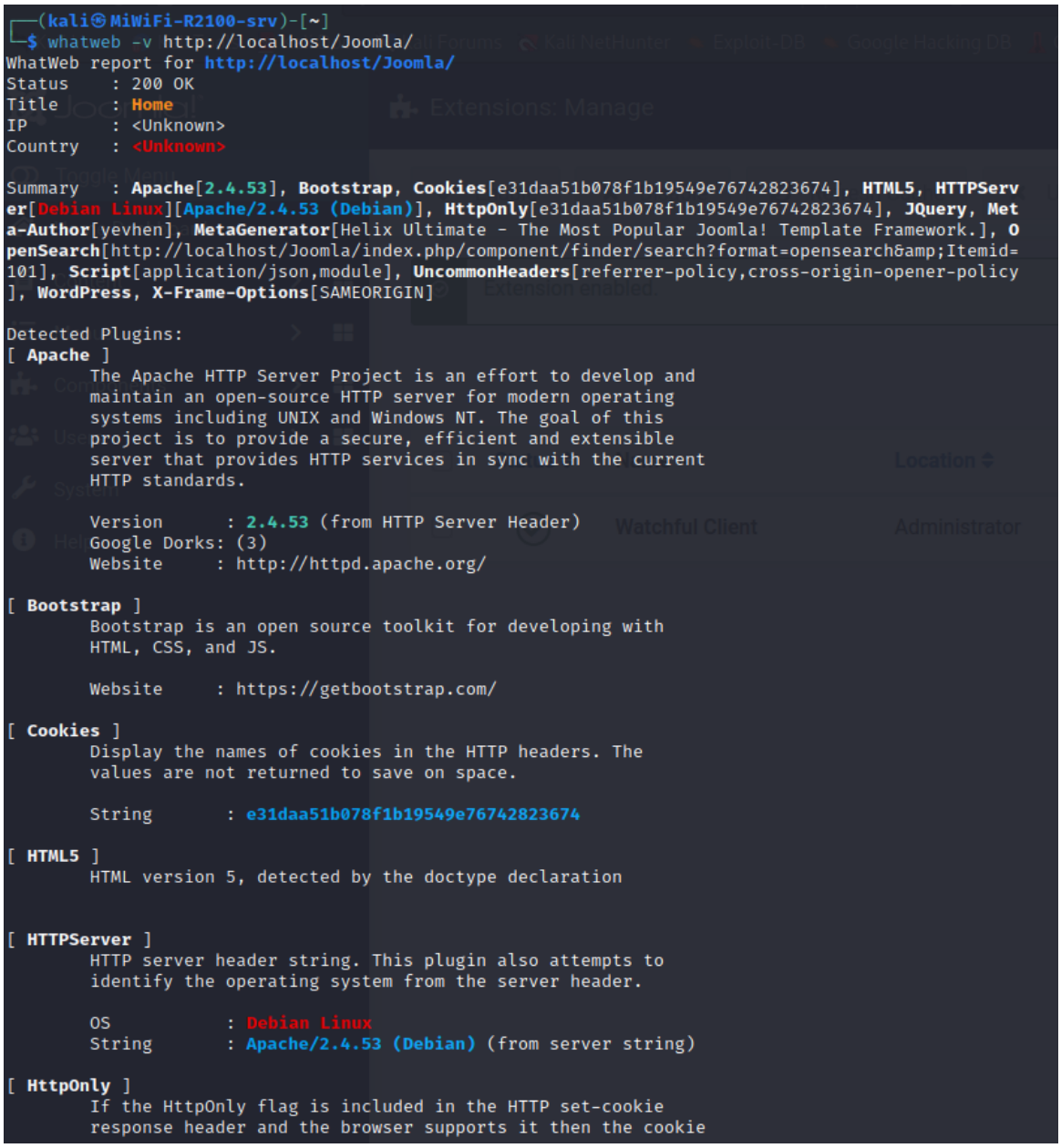

Рисунок 3.8 – Перша частина збору інформації про вебресурс з активованим захисним плагіном на базі CMS системи Joomla

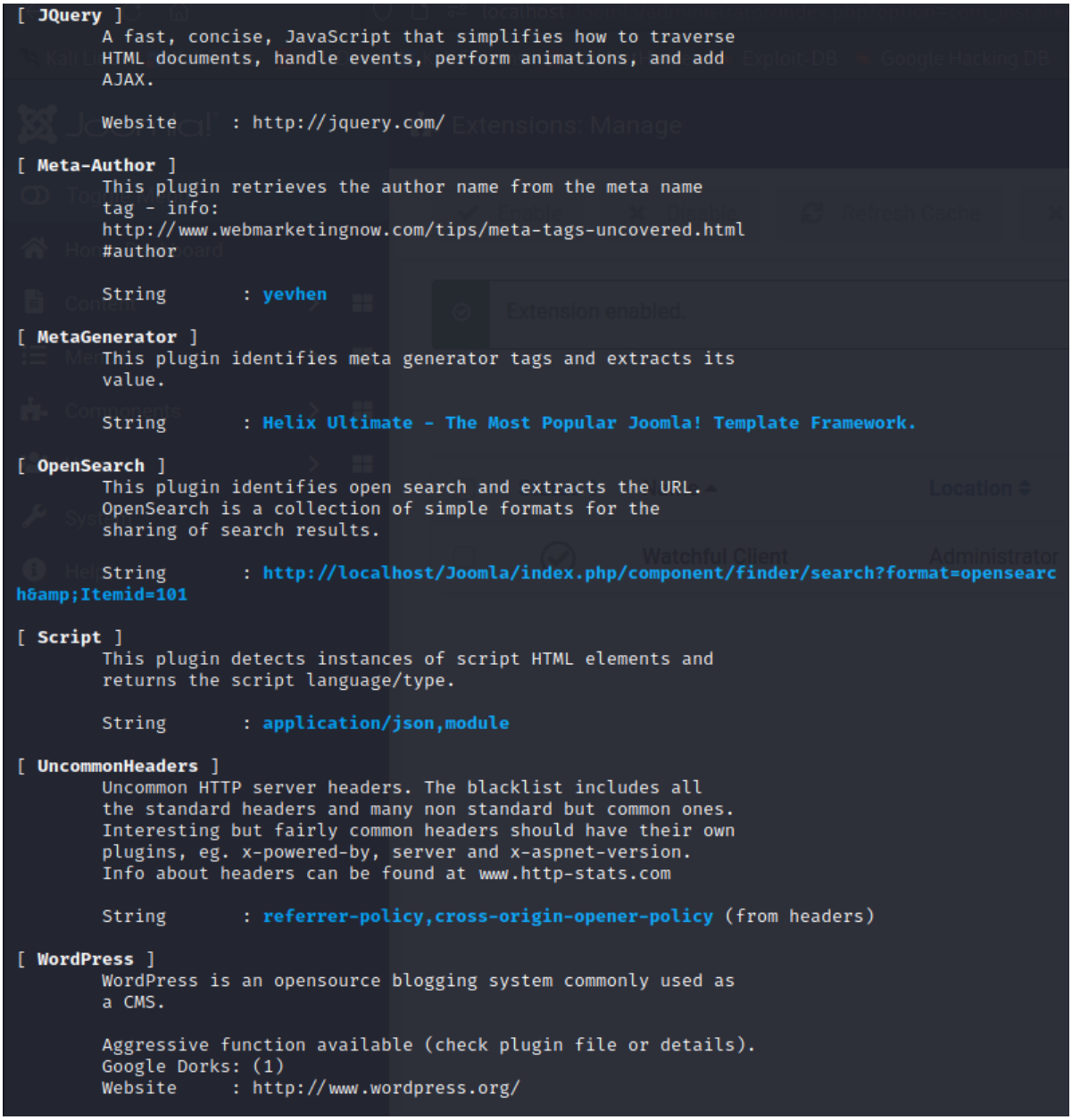

Рисунок 3.9 – Друга частина збору інформації про вебресурс з активованим захисним плагіном на базі CMS системи Joomla

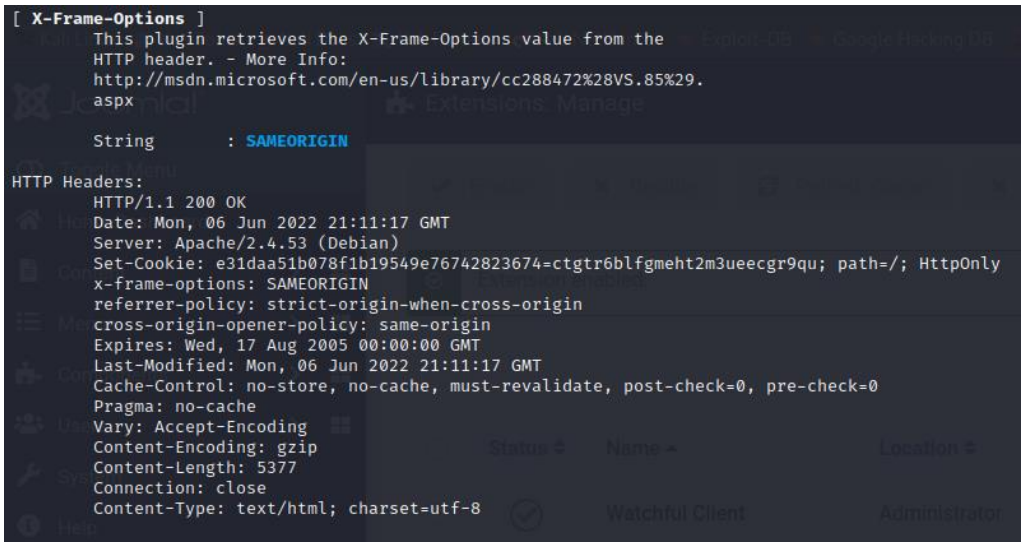

Рисунок 3.10 – Третя частина збору інформації про вебресурс з активованим захисним плагіном на базі CMS системи Joomla

Результат збору інформації щодо вебресурсу, створеного на базі CMS системи Joomla, з активованим захисним плагіном та без нього дозволяє констатувати відсутність відмінностей. Тому можна зробити висновок, що встановлений захисний плагін ніяким чином не перешкоджає збору інформації про вебресурс.

Надалі необхідно провести аналіз доступних HTTP методів. Для аналізу застосовуємо інструмент Nmap з використанням скрипта http-methods. При перевірці використаємо таку команду nmap -A -T4 --script http-methods --script-args http-methods.url-path='шлях до вебресурсу' localhost, в якій вказано назву інструменту, опція використання скрипта та цільовий ресурс.

Першим проаналізуємо вебресурс, створений на базі CMS системи WordPress. Результати тестування наведено на рис. 3.11.

Результат аналізу вебресурсу, створеного на базі CMS системи Joomla, наведено на рис. 3.12.

За результатами аналізу доступних HTTP методів вебресурсів, створених на базі CMS систем WordPress та Joomla, з активним захисним плагіном та без нього можна прийти до висновку, що плагін не впливає на доступність HTTP методів по відношенню до вебресурсів, що розгорнуті на локальному сервері.

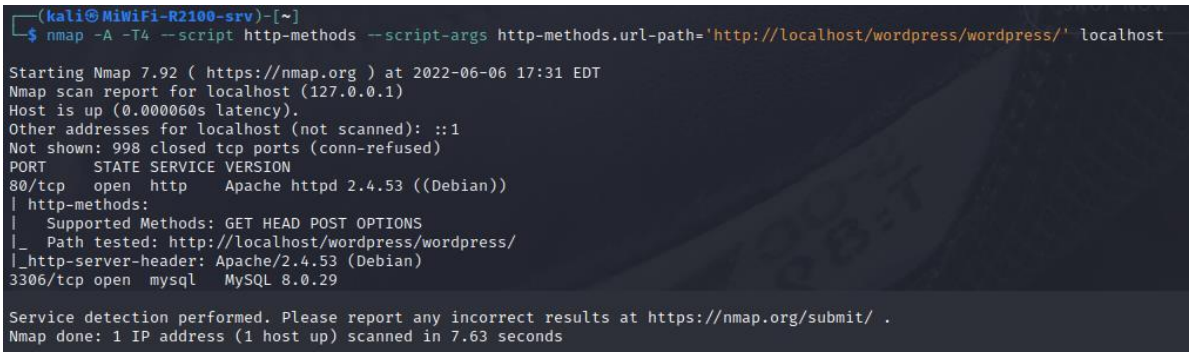

Рисунок 3.11 – Аналіз доступних HTTP методів для вебресурсу, створеного на базі CMS системи WordPress з активним захисним плагіном

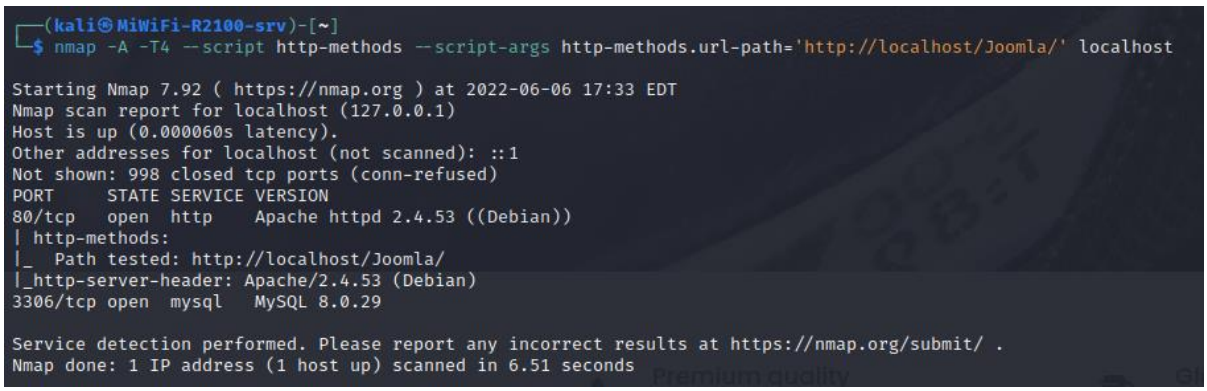

Рисунок 3.12 – Аналіз доступних HTTP методів для вебресурсу, створеного на базі CMS системи Joomla з активним захисним плагіном

Результати аналізу доступних HTTP методів збігаються для вебресурсів, які створені на різних CMS системах, через те, що ці вебресурси розгорнуті на одному локальному сервері.

Наступним етапом буде тестування на впровадження сторонніх скриптів, яке виконується за допомогою інструмента OWASP ZAP, котрий містить в собі різноманітний функціонал та має різні варіанти сканування. Для початку необхідно вибрати вид сканування (автоматизований чи ручний). Обираємо автоматизований та вводимо посилання на вебресурс. На рис. 3.13 – 3.14 наведено результати сканування вебресурсу, створеного на базі CMS системи WordPress без активного захисного плагіну, які свідчать про те, що вебресурс схильний до впровадження сторонніх скриптів.

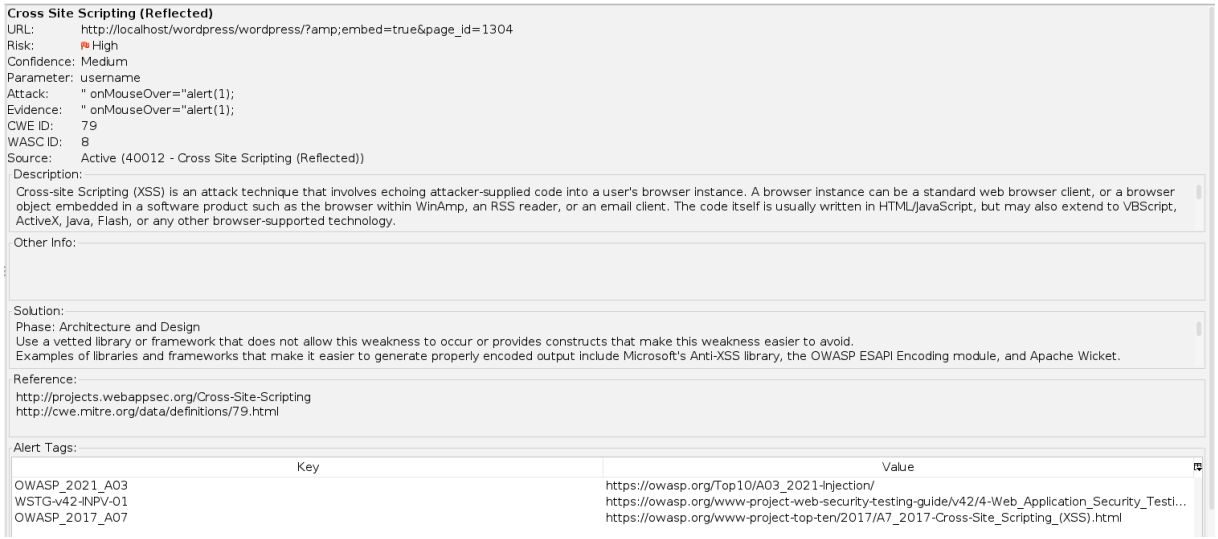

Рисунок 3.13 – Результат аналізу на можливість введення сторонніх скриптів без активного захисного плагіну

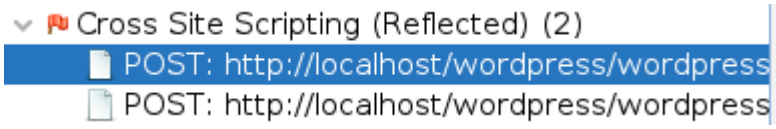

Рисунок 3.14 – Перелік загроз критичного рівня на можливість введення сторонніх скриптів без активного захисного плагіну

Проведений аналіз вебресурсу з активованим захисним плагіном за допомогою OWASP ZAP засвідчив, що цей плагін не покращує захищеність сайту від впровадження сторонніх скриптів (рис. 3.15 – 3.16).

Рисунок 3.15 – Результат аналізу на можливість введення сторонніх скриптів з активним захисним плагіном

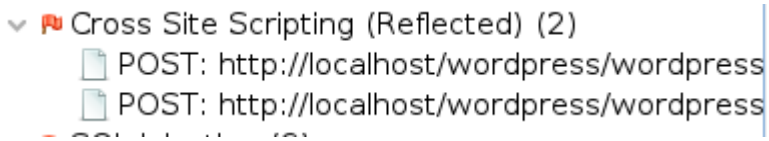

Рисунок 3.16 – Перелік загроз критичного рівня на можливість введення сторонніх скриптів активним захисним плагіном

Проведено аналогічні дії по відношенню до вебресурсу, створеного на базі CMS системи Joomla з активним захисним плагіном. Результати перевірки продемонстровано на рис. 3.17 – 3.18.

Рисунок 3.17 – Результат аналізу на можливість введення сторонніх скриптів з активним захисним плагіном

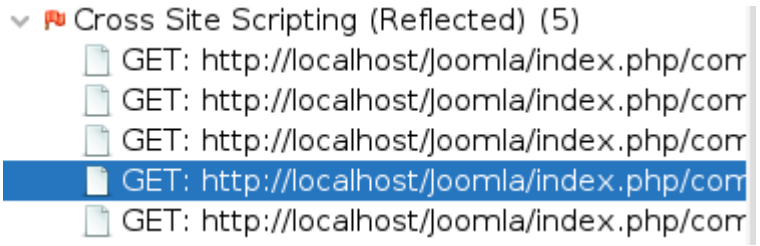

Рисунок 3.18 – Перелік загроз критичного рівня на можливість введення сторонніх скриптів активним захисним плагіном

Результати аналізу вебресурсу без активного захисного плагіну наведено на рис. 3.19 – 3.20.

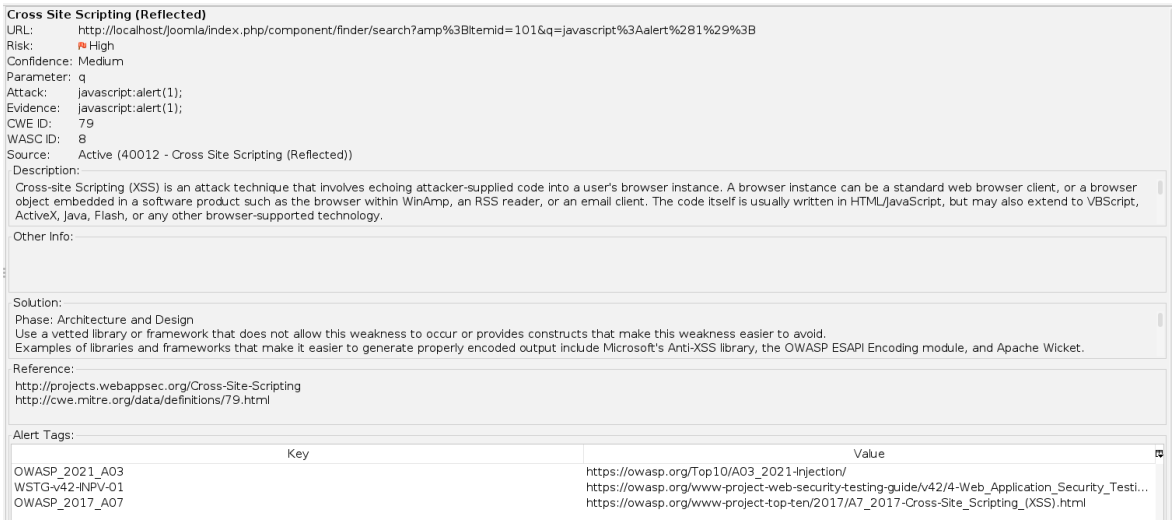

Рисунок 3.19 – Результат аналізу на можливість введення сторонніх скриптів без активного захисного плагіну

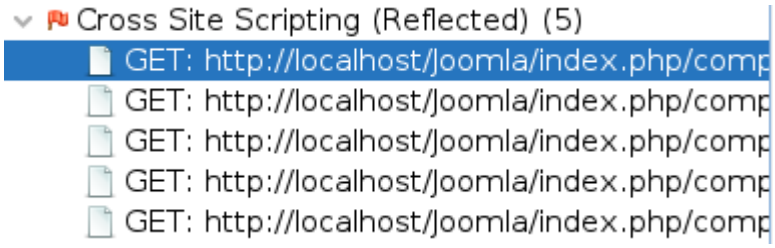

Рисунок 3.20 – Перелік загроз високого рівня на можливість введення сторонніх скриптів без активного захисного плагіну

При порівняння результатів перевірки з використанням захисного плагіну та без нього, поліпшення безпеки у відношенні до загрози введення стороннього скрипта, не було відзначено.

Тепер проведемо аналіз на можливість впровадження SQL ін’єкцій за допомогою OWASP ZAP. Під час сканування вебресурсів на захищеність від введення сторонніх скриптів також було проведено аналіз на можливість SQL ін’єкцій, було проаналізовано вебресурси з активним захисним плагіном та без нього. Результати аналізу вебересурсу, створеного на базі CMS системи WordPress без активного захисного плагіну, наведено на рис. 3.21 – 3.22.

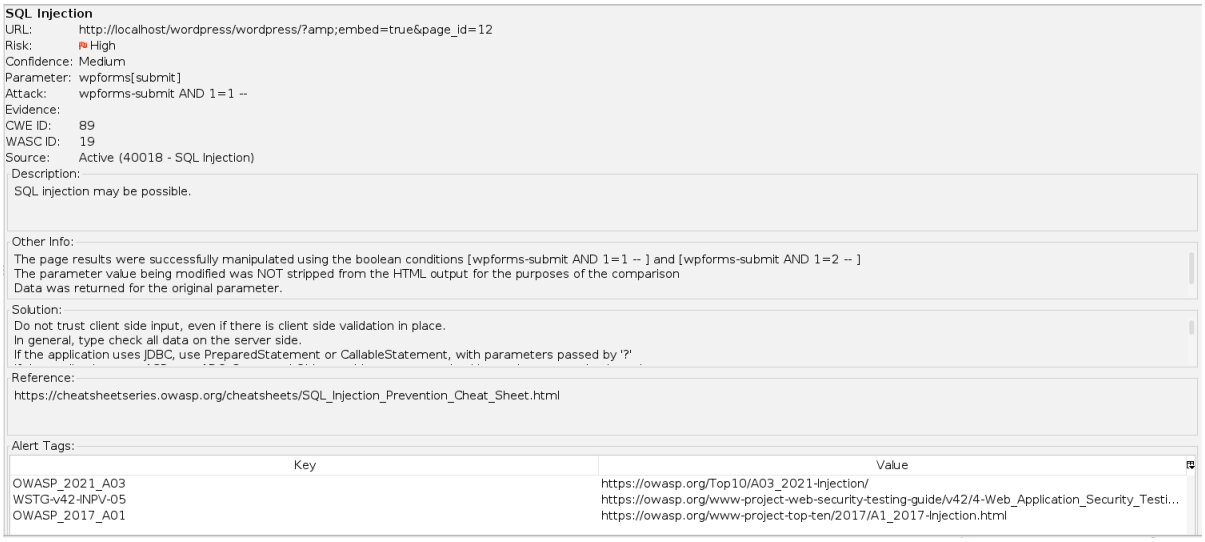

Рисунок 3.21 – Результат аналізу на можливість SQL ін’єкцій скриптів без активного захисного плагіну

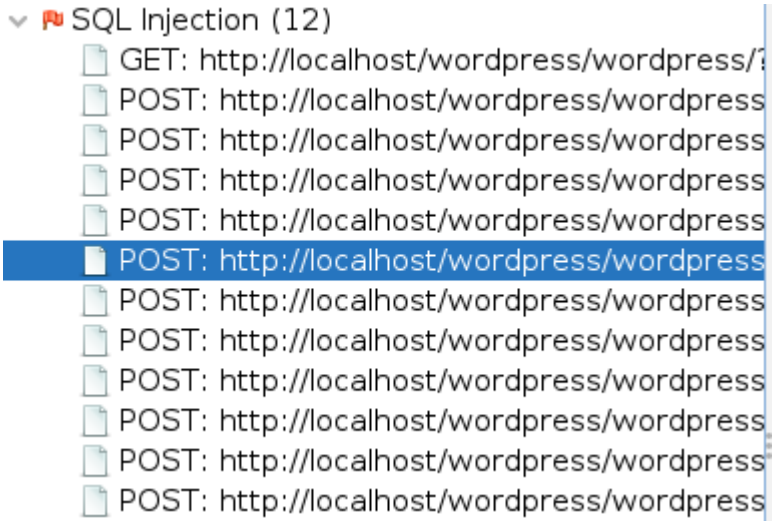

Рисунок 3.22 – Перелік загроз критичного рівня на можливість SQL ін’єкцій без активного захисного плагіну

Результати аналізу вебересурсу, створеного на базі CMS системи WordPress з активним захисним плагіном, зображено на рис. 3.23 – 3.24.

Результати аналізу вебересурсу, створеного на базі CMS системи Joomla з активним захисним плагіном, наведено на рис. 3.25 – 3.26.

Рисунок 3.23 – Результат аналізу на можливість SQL ін’єкцій з активним захисним плагіном

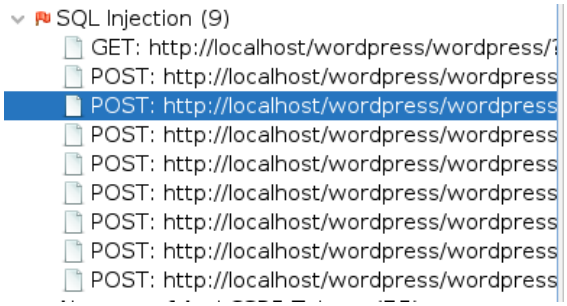

Рисунок 3.24 – Перелік загроз критичного рівня на можливість SQL ін’єкцій з активним захисним плагіном

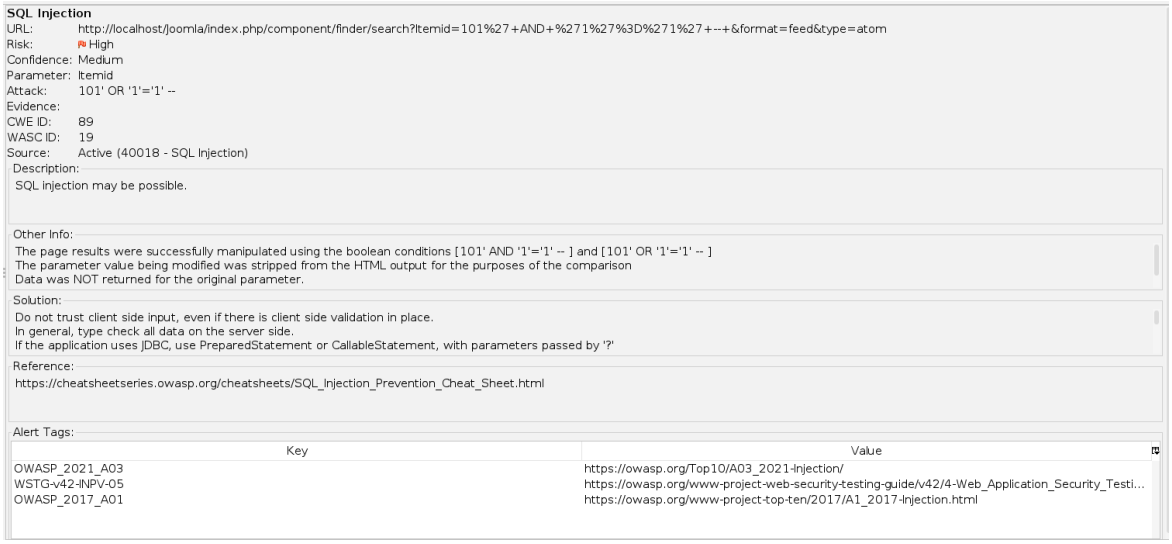

Рисунок 3.25 – Результат аналізу на можливість SQL ін’єкцій з активним захисним плагіном

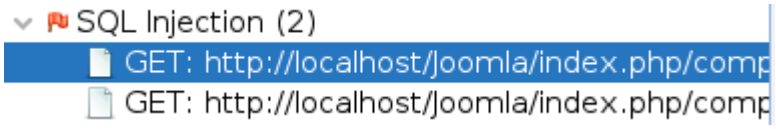

Рисунок 3.26 – Перелік загроз критичного рівня на можливість SQL ін’єкцій з активним захисним плагіном

Результати аналізу вебересурсу, створеного на базі CMS системи Joomla без активного захисного плагіну, наведено на рис. 3.27 – 3.28.

Рисунок 3.27 – Результат аналізу на можливість введення SQL ін’єкцій без активного захисного плагіну

Рисунок 3.28 – Перелік загроз критичного рівня на можливість SQL ін’єкцій без активного захисного плагіну

Порівнюючи результати аналізу двох вебресурсів, можна констатувати, що захисний плагін дозволяє знизити кількість вразливостей, пов’язаних з SQL ін’єкціями, лише у випадку вебресурсу, створеного на базі Wordpress. У той же час, як у випадку вебресурсу, створеного на базі Joomla, захисний плагін не дав ніяких результатів. Для перевірки раніше виявлених вразливостей використаємо інструмент SQLmap. Для початку аналізу вводимо команду sqlmap -u “посилання на вебресурс” --batch –banner/--password, після чого вебресурс атакується з використанням SQL ін’єкцій (рис. 3.29 – 3.30).

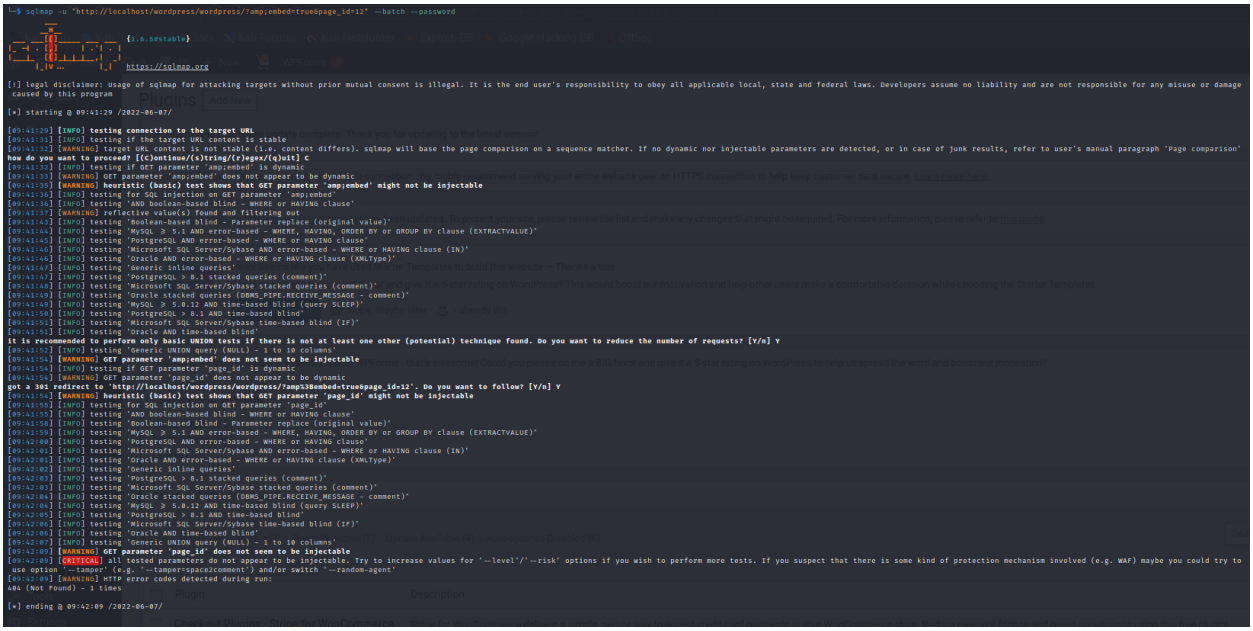

Рисунок 3.29 – Результат аналізу на SQL ін’єкції вебресурсу на базі WordPress

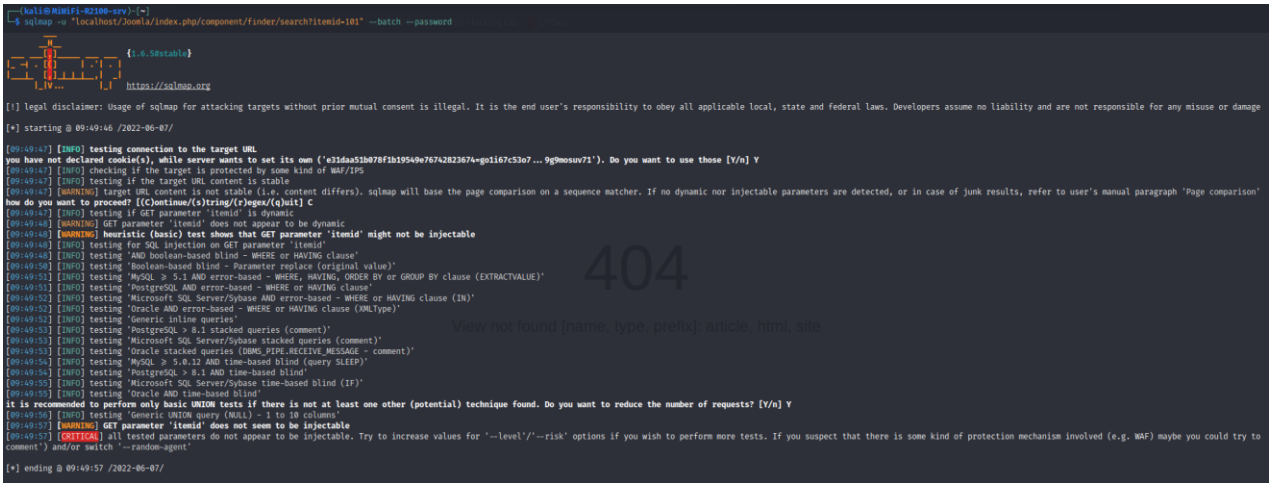

Рисунок 3.30 – Результат аналізу на SQL ін’єкції вебресурсу на базі Joomla

Аналізуючи отримані результати, SQLmap не виявив можливості на введення SQL ін’єкцій до вебресурсів, хоча при скануванні OWASP ZAP такі вразливості були ідентифіковані. Попри такі результати було проведено ще один аналіз. Була здійснена спроба атакувати знайдені вразливості за допомогою введення запропонованих ін’єкцій у вразливі місця, які було знайдено за допомогою OWASP ZAP. При виконанні запиту «http://localhost/Joomla/index.php/component/finder/search?itemid=101+AND+1%3D1+--&format=feed&q=ZAP&type=atom», у створеному вебресурсі на базі CMS системи Joomla було отримано такі результати (рис. 3.31).

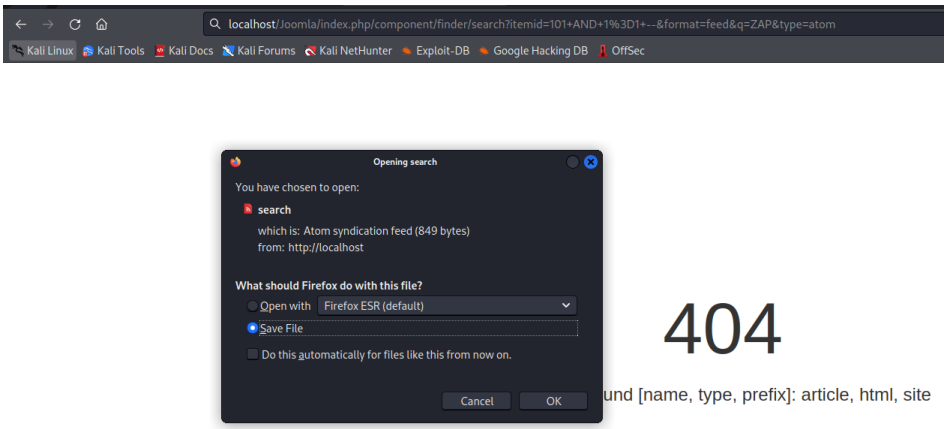

Рисунок 3.31 – Результат аналізу на SQL ін’єкції вебресурсу на базі Joomla за допомогою ручного аналізу

У результаті отримано запит на завантаження файлу під назвою «search», який містить в собі такі дані, зображені на рис. 3.32.

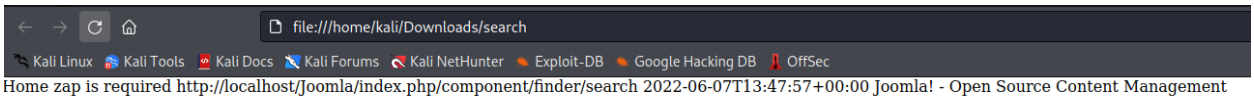

Рисунок 3.32 – Вміст файлу під назвою «search»

Як можна переконатися, жодний чутливих даних не було виявлено. Однак це підводить нас до того, що можливість проникнення дуже висока.

При спробі ручного введення SQL ін’єкцій у вебресурс, створений на базі CMS системи WordPress, таких результатів досягнуто не було.

За підсумками аналізу захищеності можна відзначити, що збір інформації про вебресурси вдалося здійснити просто та без перешкод. У результаті з’ясовано який сервер був використаний та його версію, інформацію про використані технології та CMS систему при створенні, її версію. Тестування HTTP методів показало, що доступними є такі методи: GET, POST, OPTIONS та GET. Аналіз на можливість введення сторонніх скриптів засвідчив, що така загроза існує в незалежності від увімкненого захисного плагіну. Аналіз на можливість введення SQL ін’єкцій показав, що така загроза існує, хоча і не вдалося це підтвердити за допомогою автоматизованої системи на обох

вебресурсах. Проте використовуючи ручне тестування, було підтверджено одну із виявлених вразливостей у вебресурсі, створеному на базі CMS системи Joomla.

Загальні дані аналізу відображені у Таблиці 1.

Таблиця 1 – Порівняння результатів аналізу

CMS система	WordPress з захисним плагіном	Joomla з захисним плагіном	WordPress без захисного плагіну	Joomla без захисного плагіну
Введення сторонніх скриптів	4	3	4	3
SQL ін’єкції	4	3	3	3
Ручний аналіз SQL ін’єкції	5	3	5	3

Оцінки виставлені у діапазоні від 1 до 5: 1 – не захищена, 2 - майже не захищена, 3 – середня захищеність, 4 – майже захищена, 5 - захищена.

Загалом аналіз захищеності вебресурсів, створених за допомогою CMS систем WordPress та Joomla, показав, що у цих вебресурсах доступний збір інформації, а також вдалося дізнатися, які порти використовуються та доступні методи HTTP. Аналіз на вразливість до XSS засвідчив, що створені вебресурси мають такі вразливості, а захисні плагіни не допомагають їх позбутися. Однак деякі вразливості, які пов’язані з SQL ін’єкціями, плагіни Jetpack та Watchful змогли заблокувати.

ВИСНОВКИ

У випускній роботі досліджено основні вразливості систем керування вмістом та розроблено методику аналізу захищеності вебресурсів, створених за допомогою CMS систем. Для проведення аналізу захищеності створено вебресурси на базі CMS систем WordPress та Joomla. Основою для розробки методики дослідження стала існуюча методологія тестування безпеки вебресурсів OWASP Web Security Testing Guide, з якої обрано окремі етапи. Для обраних етапів здійснено аналіз та опис автоматизованих і ручних методів аналізу захищеності, розглянуто спеціалізований інструментарій, за допомогою якого можливо проаналізувати захищеність вебресурсу. Зокрема, для аналізу обрано такі інструменти, як WhatWeb, Nmap, OWASP ZAP та SQLmap. Кожен з цих інструментів має певні функціональні можливості та відповідає за конкретний етап.

Проведений аналіз захищеності вебресурсів, створених на базі CMS систем WordPress та Joomla, показав, що при достатньому рівні захисту певні недоліки в захищеності ресурсів все ж таки існують. Зважаючи на результат аналізу, встановлено, що вебресурси, які розроблено з використанням певних плагінів (Starter Template, WooCommerce, Jetpack) для WordPress та (HikaShop, Watchful) для Joomla, мають середній рівень захищеності. Визначено основні вразливості вебресурсів, створених на базі CMS систем WordPress та Joomla. При цьому, треба враховувати, що аналіз здійснювався в «стерильних» умовах на локальному сервері, що могло вплинути на результати аналізу. Однак, загалом виявлені загрози не є критичними, тому створені вебресурси можна вважати безпечними для користувачів.

Аналіз підтвердив факт того, що використання систем керування вмістом є доцільним через простоту створення вебресурсів та вбудований захист для користувачів без спеціальних навичок розроюлення вебресурсів.

СПИСОК ЛІТЕРАТУРИ

1. This CMS cyberattack has affected thousands of sites worldwide [Electronic resource]. – Access mode: https://www.techradar.com/news/this-cms-cyberattack-has-affected-thousands-of-sites-worldwide
2. Злам API Drupal [Electronic resource]. – Access mode: https://apisecurity.io/issue-20-drupal-apis-hacked-eu-iot-standards/
3. How to Secure Your Content Management System (CMS) [Electronic resource]. – Access mode: https://securityboulevard.com/2019/08/how-to-secure-your-content-management-system-cms/
4. Что такое WordPress [Electronic resource]. – Access mode: https://www.hostinger.com.ua/rukovodstva/chto-takoe-wordpress-obzor-populjarnoj-cms/
5. Основи: Drupal як система управління сайтом (CMS) [Electronic resource]. – Access mode: https://www.drupal.org/ru/docs/user_guide/ru/understanding-drupal.html
6. LAMP [Electronic resource]. – Access mode: https://uk.wikipedia.org/wiki/LAMP
7. Joomla [Electronic resource]. – Access mode: https://joomla.ru/docs/articles/2676-five-reasons
8. CMS Security: How to Keep Your Website Safe [Electronic resource]. – Access mode: https://www.brandextract.com/Insights/Articles/CMS-Security-How-to-Keep-Your-Website-Safe/
9. CMS Vulnerabilities: Why Are CMS Platforms Common Hacking Targets [Electronic resource]. – Access mode: https://beaglesecurity.com/blog/article/cms-vulnerabilities.html
10. How Hackers Exploited your WordPress Website in 2018[Electronic resource]. – Access mode: https://www.getastra.com/blog/cms/wordpress-

[Electronic resource]. – Access mode: security/most-common-wordpress-attacks/

11. 5 Most Critical Vulnerabilities Ever Found on Drupal[Electronic resource]. – Access mode: https://www.getastra.com/blog/cms/drupal-vulnerability-5-most-critical-vulnerabilities-ever-found-on-drupal/
12. 3 Most Common Vulnerabilities Found in Joomla [Electronic resource]. – Access mode: https://www.getastra.com/blog/cms/joomla-security/joomla-vulnerabilities-3-most-common-vulnerabilities-found/
13. Starter Templates — Elementor, WordPress & Beaver Builder Templates [Electronic resource]. – Access mode: https://wordpress.org/plugins/astra-sites/
14. WooCommerce [Electronic resource]. – Access mode: https://wordpress.org/plugins/woocommerce/
15. Jetpack [Electronic resource]. – Access mode: https://wordpress.org/plugins/jetpack/
16. HikaShop[Electronic resource]. – Access mode: https://www.hikashop.com/
17. Watchful[Electronic resource]. – Access mode: https://watchful.net/
18. Nikto2[Electronic resource]. – Access mode: https://cirt.net/Nikto2
19. OWASP ZAP [Electronic resource]. – Access mode: https://www.zaproxy.org/docs/
20. SQLmap [Electronic resource]. – Access mode: https://sqlmap.org/
21. Nmap [Electronic resource]. – Access mode: https://nmap.org/
22. Vega Scanner [Electronic resource]. – Access mode: https://subgraph.com/vega/
23. WPScan [Electronic resource]. – Access mode: wpscan.com
24. WhatWeb [Electronic resource]. – Access mode: https://github.com/urbanadventurer/WhatWeb
25. OWASP WSTG - v4.2 [Electronic resource]. – Access mode: https://owasp.org/www-project-web-security-testing-guide/v42/