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 магістра

студента *Шестерніної Оксани Леонідівни*

академічної групи *125м-19-2*

напряму підготовки *125 Кібербезпека*

спеціалізації[1]

за освітньо-професійною програмою *Кібербезпека*

на тему *Методика тестування та виявлення SQL-ін'єкцій для веб-сайтів на CMS Magento*

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	проф., д.ф.-м.н., Кагадій Т.С			
розділів:				
спеціальний	ас. Мілінчук Ю.А.			
економічний	к.е.н., доц. Пілова Д.П.			
Рецензент				
Нормоконтроль	ст. викл. Тимофєєв Д.С.			

Дніпро
2020

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

«_____»_____20___року

ЗАВДАННЯ
на кваліфікаційну роботу ступеня бакалавра

студенту *Шестерніній Оксані Леонідівні* **академічної групи** *125м-19-2*
(прізвище та ініціали) (шифр)

спеціальності *125 Кібербезпека*

спеціалізації

за освітньо-професійною програмою *Кібербезпека*

на тему *Методика тестування та виявлення SQL-ін'єкції для веб-сайтів на CMS Magento*

Затверджену наказом ректора НТУ «Дніпровська політехніка» від 22.10.2020 № 888-с

Розділ	Зміст	Термін виконання
Розділ 1	Огляд джерел за темою та напрямом дослідження Провести аналіз нормативно-правової бази у сфері захисту інформації	28.10.20
Розділ 2	Проведення дослідження принципу роботи SQL-ін'єкцій. Розробка методики тестування та виявлення SQL-ін'єкцій для веб-сайтів на CMS Magento	18.11.20
Розділ 3	Провести розрахунок вартості розроблення методики тестування та виявлення	02.12.20

Завдання видано _____ проф., д.ф.-м.н., Кагадій Т.С
(підпис керівника) (прізвище, ініціали)

Дата видачі завдання: **12.09.2020р.**

Дата подання до екзаменаційної комісії: **8.12.2020р.**

Прийнято до виконання _____ ___Шестерніна О.Л.__
(підпис студента) (прізвище, ініціал

РЕФЕРАТ

Пояснювальна записка: 89 сторінок, 2 таблиці, 20 рисунків, 9 додатків, 22 джерела.

Об’єкт дослідження: SQL-ін‘єкція.

Предмет розробки: тестування та виявлення SQL ін’єкцій для веб-сайтів розроблених на CMS Magento.

Мета кваліфікаційної роботи: розробити методику тестування та виявлення SQL ін’єкцій для веб-сайтів розроблених на CMS Magento.

Методи дослідження: аналіз, спостереження, порівняння, оцінка.

У першому розділі було розглянуто поняття CMS системи, її основне призначення та особливості. Основна увага у розділі була приділена на технічні особливості та різницю у характеристиках основних CMS систем.

У спеціальній частині роботи були виділені та розглянуті основні вразливості сайтів розроблених на різних CMS системах які найчастіше стають причиною злому сайту. Основна увага була приділена SQL-ін’єкціям їх принципу роботи та особливостями. У роботі була розроблена та запропонована методика тестування та виявлення SQL-ін’єкцій для веб-сайтів розроблених за допомогою CMS Magento. Розроблена пошагова інструкція, шаблони тестових сценаріїв та шаблони звітів які необхідно використовувати при проведенні тестування сайту.

В економічній частині роботи було розраховано витрати на впровадження розробленої методики та загальні збитки інтернет-магазину від реалізації SQL атак у рік.

Практична цінність полягає у можливості використання розробленої методики для аналізу та тестування виявлення SQL-ін'єкцій для веб-сайтів розроблених на CMS Magento.

МЕТОДИКА ТЕСТУВАННЯ ТА ВИЯВЛЕННЯ, МЕТОДИКА ТЕСТУВАННЯ, CMS СИСТЕМА, CMS MAGENTO, SQL-ІН’ЄКЦІЯ, БАЗА ДАНИХ, ВРАЗЛИВОСТІ САЙТІВ.

РЕФЕРАТ

Пояснительная записка: 89 страниц, 2 таблицы, 20 рисунков, 9 приложений, 22 источника.

Объект исследования: SQL-инъекция.

Предмет разработки: тестирование и выявление SQL-инъекций для веб-сайтов на CMS Magento.

Цель квалификационной работы: разработать методику тестирования и выявления SQL-инъекций для веб-сайтов разработанных на CMS Magento

Методы исследования: анализ, наблюдение, сравнение, оценка.

В первом разделе рассмотрено понятие CMS системы, ее основное предназначение и особенности. Основное внимание было уделено на технические особенности и разницу в характеристиках основных CMS систем.

В специальной части работы были выделены и рассмотрены основные уязвимости сайтов разработанных на разных CMS системах, которые чаще всего стают причиной взлома сайта. Основное внимание было уделено SQL-инъекциям их принципу работы и особенностям. В работе была разработана и предложена методика тестирования и выявления SQL-инъекций для веб-сайтов разработанных на CMS Magento. Разработана пошаговая инструкция, шаблоны тестовых сценариев и шаблоны отчетов, которые необходимо использовать при проведении тестирования сайта на CMS Magento.

В экономической части работы были просчитаны расходы на внедрение разработанной методики и общие убытки интернет-магазина от реализации SQL атак в год.

Практическая ценности заключается в возможности использования разработанной методики для сайта или интернет-магазина на CMS Magento.

МЕТОДИКА ТЕСТИРОВАНИЯ И ВЫЯВЛЕНИЯ, МЕТОДИКА ТЕСТИРОВАНИЯ, CMS СИСТЕМА, CMS MAGENTO, SQL-ИНЪЕКЦИЯ, БАЗА ДАННЫХ, УЯЗВИМОСТИ САЙТОВ.

ABSTRACT

Explanatory Note: 89 pages, 2 tables, 20 pictures, 9 applications, 22 sources.

Object of research: SQL injection.

Subject of development: testing and detecting SQL injections for CMS Magento websites.

The purpose of the thesis: Develop a methodology for testing and detecting SQL injections for websites developed on CMS Magento.

Methods: analysis, observation, comparison, assessment.

The first section described the concept of a CMS system, its main purpose and features. The main focus was on the technical features and the difference in the characteristics of the main CMS systems.

In a special part of the work, the main vulnerabilities for sites were identified and considered. The described vulnerabilities most often become the reasons for hacking a site. The main focus in the section was on SQL injection, in particular, how it works and descriptions. The methodology for testing and detecting SQL injections for websites developed on CMS Magento was developed and proposed. Step-by-step instructions, test case templates and report templates were given in the work. This documentation can be used and will be helpful during testing of the website on CMS Magento.

In the economic part of the work, the costs of implementing the developed methodology and the total losses of the online store from the implementation of SQL attacks per year were calculated.

The practical value lies in the possibility of using the developed methodology for a website or online store.

TESTING AND DETECTION TECHNIQUE, TESTING TECHNIQUE, CMS SYSTEM, CMS MAGENTO, SQL-INJECTION, DATABASE, SITES VULNERABILITY.

СПИСОК УМОВНИХ СКОРОЧЕНЬ

CMS - система управління вмістом;

CVE - безкоштовна база даних про вразливість систем безпеки;

XSS (англ. Cross-Site Scripting) - міжсайтовий скриптинг;

CMA - Додаток для управління вмістом;

CDA - Додаток контенту;

CRM (Customer Relationship Management) - набір інструментів для управління клієнтами;

OWASP (Open Web Application Security Project) - Інтернет-спільнота, яка виробляє статті, методології, документацію, інструменти та технології в галузі безпеки веб-додатків;

ПК - персональний комп'ютер;

БД - база даних;

НД ТЗІ – нормативний документ технічного захисту інформації;

URL – Uniform Resource Locator;

HTML – HyperText Markup Language.

ЗМІСТ

ВСТУП

Сучасний світ вимагає від нас піклуватись не тільки про свою безпеку, а й про безпеку веб-сайтів. Це стосується як і власників бізнесу, так і користувачів, бо обидві сторони можуть стати жертвами кіберзлочинців.

Веб-сайти займають значну роль у сучасному Інтернеті та житті користувачів, основними функціями яких є інформативність та доступність, іншими словами вони представляють інтереси особи, компанії чи організації.

Оскільки веб-сайти доступні з будь-якої точки земної кулі, де є вихід в Інтернет, тема їх безпеки доволі актуальна, бо сайти знаходяться під пильним поглядом кіберзлочинців, які тільки й чекають аби заволодіти персональними даними користувачів.

Постає питання, як же правильно та надійно захистити свій сайт та не втратити довіру клієнтів.

По-перше, потрібно ще з моменту ідеї розробки сайту підійти дуже ретельно до вибору CMS системи та рівня безпеки для його реалізації.

По-друге, не потрібно нехтувати послугами спеціалістів з кібербезпеки та обрати правильний алгоритм захисту сайту.

Спираючись на статистику, більшість зломів пов'язані не з вразливостями в самих CMS, а з неправильною конфігурацією, а також вразливостях в плагінах і темах, які адміністратори часто забувають оновити. Так, лише 56% вивчених сайтів працюють з актуальними версіями програмного забезпечення, саме це і призводить до “легкого” злому та крадіжки персональних даних [7].

Якщо розглянути один з найпоширеніших шляхів злому веб-сайту, то це, як правило, SQL ін’єкція, бо зловмисники хочуть заволодіти саме базою даних та втрутитись у базу даних сайту. Якщо сайт недостатньо захищений, то кіберзлочинцю буде не тільки просто втрутитись в базу даних сайта, але й змінити налаштування, видалити записи, або зміни запити в базі даних.

Зараз в Інтернеті існує тисячі вірус-ботів, які сканують мільйони сайтів на день, та зламують інтернет ресурси. З огляду на продуктивність, сканування при

недостатній захищеності системи - злом самого вашого сайту це тільки питання часу, саме тому методика тестування та виявлення вразливостей для веб-сайтів зараз дуже актуальна та просто необхідна.

Метою кваліфікаційної роботи є розробка методики тестування та виявлення SQL ін'єкцій для веб-сайтів розроблених на CMS Magento.

Об'єкт дослідження у даній роботі: SQL-ін'єкція.

Предмет розробки: тестування та виявлення SQL ін'єкцій для веб-сайтів розроблених на CMS Magento.

РОЗДІЛ 1. СТАН ПИТАННЯ. ПОСТАНОВКА ЗАДАЧІ

1.1 Стан питання

У 2020 році важко знайти бізнес будь-якого розміру, який би не мав особистого веб-сайту, розробленого за допомогою CMS системи чи самописно. Або ж бізнес, який би не використовував інформаційні технології щодня хоча б для одного з аспектів управління компанією. Ось чому кібербезпека стала такою важливою проблемою в сьогоднішньому світі бізнесу.

Загроза кібератак не тільки зростає з кожним роком, але й потенційний вплив цих атак стає все більш серйозним, оскільки технологічний прогрес і більший відсоток ділових угод розгортаються в цифровому світі. Користувачі залишають все більше інформації на веб-сторінках, роблять все більш замовлень та реєструються на різних веб-ресурсах. На жаль, не завжди наші дані зберігаються достатньо захищено та правильно, особливо, коли це стосується веб-сайтів розроблених саме за допомогою CMS систем.

Крім того, недавні дослідження безпеки свідчать про те, що більшість компаній мають незахищені дані та погану практику кібербезпеки, що робить їх уразливими до втрати даних. Так прогнозується, що світові витрати на кібербезпеку сягнуть 133,7 мільярда доларів у 2022 році [3].

Зростаючий обсяг масштабних, широко розрекламованих порушень говорить про те, що не тільки зростає кількість порушень безпеки - вони також збільшуються по важкості. Порушення даних викриває конфіденційну інформацію, яка часто піддає схильним користувачам ризик крадіжки особистих даних, руйнує репутацію компаній і майже завжди залишає компанію відповідальною за порушення правил.

Що стосується кібербезпеки, то не всі галузі є рівними з точки зору рівня захищеності та необхідних правил захищеності. Галузі, які зберігають цінну інформацію, такі як охорона здоров'я та фінанси, зазвичай є більшою ціллю для хакерів, які хочуть викрасти номери соціального страхування, медичні записи та інші особисті дані. Тому вони враховують усі особливості захисту таких систем. Але насправді ніхто не є у безпеці у цій ситуації, оскільки інші галузі з меншим

ступенем ризику також є під прицілом кіберзлочинців через уявлення, що у них буде менше заходів безпеки.

Враховуючи всі дані, що були зазначені вище, необхідність чіткого розуміння та надійного вибору CMS системи для свого бізнесу постає на перший план. Для того, щоб розібратися яка ж система найбільш надійна, потрібно витратити немало часу і залучити спеціалістів з кібербезпеки, бо 50% компаній, які були жертвами кіберзлочинців зачиняються і витрачають величезні суми на усунення збитків атак, які можна було б не допустити встановив необхідні налаштування безпеки.

1.1.1 CMS система та її призначення

Система управління вмістом, яку часто скорочують як CMS - це програмне забезпечення, яке допомагає користувачам/власникам бізнесу створювати, керувати та модифікувати вміст на веб-сайті без необхідності у спеціальних технічних знаннях [3].

Якщо говорити просто, система управління вмістом - це інструмент, який допомагає вам створити веб-сайт у якому не потрібно писати весь код з нуля (або навіть взагалі знати, як писати код).

Сайт розроблений за допомогою CMS спрощує роботу контент-менеджерам і іншим фахівцям, які будуть працювати з сайтом у майбутньому [1].

На більш технічному рівні система управління вмістом складається з двох основних частин:

- Додаток для управління вмістом (CMA) - це частина, яка дозволяє вам фактично додавати та керувати контентом на вашому сайті.
- Додаток контенту (CDA) - це внутрішній, технічний процес, який бере вміст, який ви вводите в CMA, зберігає його належним чином і робить його доступним для відвідувачів.

Разом дві системи спрощують підтримку веб-сайту та працюють як один злагоджений механізм.

Також великим плюсом є те, що сучасні CMS системи досить гнучкі та дозволяють розробити веб-сайт, що буде відповідати необхідним вимогам.

Більшість популярних CMS систем можуть бути використані для створення по суті будь-якого типу веб-сайту.

Кожен власник бізнесу стикався з необхідністю створення сайту для своєї компанії, а потім з проблемою вибору CMS системи, яких зараз на ринку існує дуже багато.

Тому перш ніж почати обирати та порівнювати CMS платформ, ось що вам треба шукати у хорошій системі.

По-перше, це простота використання, адже вам потрібна система управління вмістом, яка полегшує вам створення та редагування контенту. Це часто означає наявність інтерфейсу для швидкого додавання та видалення різних контент елементів і оновлення сторінок. Ви повинні мати змогу швидко та просто внести зміни до контенту на своєму сайті після його публікації.

По друге, це варіанти оформлення. Ваше програмне забезпечення для управління контентом має пропонувати вам безліч шаблонів дизайну веб-сайтів на вибір. Це також повинно дозволити вам легко налаштувати та змінювати дизайн сайту відповідно до власних вимог (в ідеалі це потрібно бути можливим без написання додаткового коду).

По-третє, це легке експортування даних. Якісна CMS платформа повинна мати інструменти для того, щоб ви могли легко експортувати дані та розмістити/перенести їх в інше місце. Наприклад, через деякий час ви вирішите змінити платформу або обрати іншу хостингову компанію. Вбудована можливість експортування даних полегшує вам перехід на іншу систему та дозволяє легко керувати даними.

Ще однією важливою функцією є можливість розширення та доповнення. Всі веб-сайти абсолютно різні, тому неможливо, щоб одна платформа управління контентом мала всі функції, які відповідали б вимогам кожного веб-сайту. Додаткові розширення та доповнення вирішують цю проблему. Це окреме програмне забезпечення, яке ви можете просто встановити на своє програмне забезпечення системи управління вмістом, щоб розширити його функції та додати

нові, коли це потрібно. Простіше кажучи, це додаткові програми для вашої CMS платформи [3].

При виборі системи, ви повинні звертати увагу на довідку та варіанти підтримки після встановлення CMS. Звичайно, основне призначення CMS системи - можливість зробити веб-сайт якомога простішим, але у вас все одно можуть виникнути питання які будуть потребувати швидкого вирішення і швидкої відповіді від розробників системи.

Якщо зробити висновок, то найпоширеніші особливості та можливості хорошої системи управління вмістом повинні включати:

- Редактор WYSIWYG (що-ти-бачиш-те-що-отримуєш);
- Шаблони сторінок;
- Бібліотека тем веб-сайтів;
- Версії та архівування вмісту;
- Мобільна оптимізація та адаптивний дизайн;
- Процес публікації;
- Формування форми;
- Планування вмісту;
- Управління активами (зображення, статті тощо);
- Кешування сторінок (або інші функції для пришвидшення доставки вмісту на сайт);
- Підтримка SEO;
- Інструменти для позначення вмісту та створення таксономії;
- Сумісність браузера;
- Масове управління;
- Функціональність електронної комерції (тобто каталог, кошик для покупок, обробка платежів);
- Управління спільнотою (тобто коментування, профілі);
- Локалізація / регіоналізація з багатомовним вмістом;
- Інші варіанти персоналізації;
- Ролі та дозволи користувачів;

- API;
- Інструменти аналітики;
- Імпорт / експорт вмісту.

Наявність хоча б більшої з перерахованих функцій допоможе вам користуватися системою довгостроково та безпечно.

1.1.2 Види CMS систем

Зараз існує багато відкритих комерційних інструментів, які можна класифікувати як CMS. Але все ж виділяють основні 10, які займають основну частину ринку [2].

На рисунку 1.1 зображено проценте відношення популярності CMS систем у користувачів по всьому світу.

Як можна побачити на рисунку, то WordPress займає близько 63,3% частки ринку, а досліджувана у роботі Magento - 1,4%.

На перший погляд вони всі такі різні та кожен має свої переваги та недоліки, але якщо розібратися більш детально виявиться, що системи мають однакову структуру, архітектуру та основні функції.

Тому розглянемо більш детально кожну с основних CMS систем для розуміння їх особливостей, можливостей та переваг. Основна увага буде приділена CMS Magento.

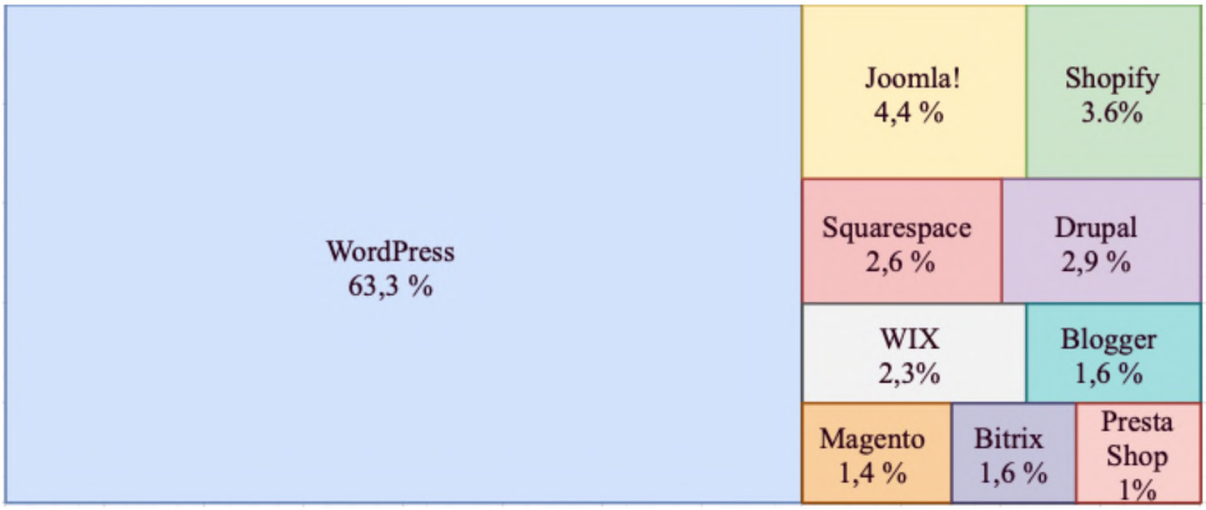


Рисунок 1.1 - 10 найпопулярніших CMS систем

Почнемо з найпопулярнішої – WordPress.org. Саме ця система вважається найпопулярнішою системою управління контентом. Хоча існують, звичайно, інші

системи, але WordPress підтримує понад 63,3% частки ринку веб-сайтів. WordPress.org – це безкоштовна система управління вмістом з відкритим кодом. Спочатку система була розроблена для ведення блогів, але зараз вона використовується для усіх видів веб-сайтів та Інтернет-магазинів.

Joomla – це ще одна популярна безкоштовна платформа CMS з відкритим кодом. Вона має безліч різних шаблонів дизайну та додаткових розширень. Нею можна користуватися безкоштовно, але вам знадобиться хостинг та доменне ім'я. Вперше CMS система була випущена у 2005 році, тому, як і WordPress, вона вдосконалюється вже не один рік. Joomla має багато функцій та має функцію встановлення веб-хостів в один клік. Однак це ідеальна платформа CMS для розробників та досвідчених розробників веб-сайтів. Початківцям буде дуже складно працювати та розібратися швидко з системою.

Drupal – це ще одна платформа CMS із відкритим кодом. Це хороший варіант для розробників або для людей, які можуть найняти розробника. Drupal підійде для тих, хто прагнете створити веб-сайт, який має багато можливостей та який повинен обробляти багато даних щодня.

WooCommerce – це найпопулярніша платформа електронної комерції у світі. З нею дійсно гнучко та легко працювати. WooCommerce технічно не є платформою CMS. Натомість вона працює як плагін для WordPress, тому вам потрібно мати WordPress на своєму сайті, щоб встановити додаток WooCommerce. Якби це була окрема платформа CMS, за даними W3Techs, вона мала б 5,8% частки ринку. Це відсоток усіх веб-сайтів у світі, які ним користуються [2].

Shopify – це універсальна платформа управління вмістом. Вам не потрібно буде купувати хостинг, встановлювати будь-яке програмне забезпечення або керувати такими речами, як оновлення та резервні копії вашої системи. Shopify має простий інтерфейс оновлення даних. Він підтримує розпродажі в магазинах, що чудово, якщо у вас є фізичний магазин, а також інтернет-магазин.

Bitrix24 – це бізнес-інструмент, який також пропонує платформу CMS поряд з іншими функціями, такими як можливість керувати своїми завданнями, проектами, комунікаціями та відносинами з клієнтами. Система безкоштовна на

базовому рівні та пропонує до 5 Гб онлайн-сховища та 12 облікових записів користувачі. Також система пропонує комплексне рішення для малого бізнесу. Якщо вам потрібен не тільки сайт, але й інструмент CRM (Customer Relationship Management), Bitrix24 може бути хорошим вибором.

Звичайно, це не все CMS системи, а лише найпопулярніші. Точної кількості CMS системи невідомо, адже кожного дня з'являються нові і нові платформа, основна мета яких – спростити та зробити доступним процесс створення сайту для кожного. Для детального дослідження у роботі була обрана CMS Magento і це не випадково.

Magento – це потужна платформа електронної комерції з відкритим кодом від величезної програмної компанії Adobe. Існує безкоштовна версія, яку ви можете завантажити та встановити у власному обліковому записі веб-хостингу під назвою Magento Open Source.

Як кажуть, Magento – це новий сценарій електронної комерції, CMS з відкритим кодом, який використовує бази даних MySQL. Magento пропонує власникам магазинів контроль за зовнішнім виглядом, змістом та функціональністю магазинів електронної комерції.

Magento CMS оснащений прекрасно розробленим адміністраторським інтерфейсом, надійним управлінням каталогами та маркетинговими інструментами. Він також сприяє великій оптимізації пошукових систем, пропонуючи можливість налаштувати сайт залежно від потреб бізнесу. Навіть з точки зору мерчандайзингу та просування, він пропонує розширений контроль. Magento CMS доступний у різних версіях, щоб допомогти компаніям належним чином обрати правильний вибір.

Magento вважається однією з найбільш функціональних платформ та найбільш налаштованою платформа електронної комерції. Також, Magento є однією з найбільш захищених та доступних для самостійного налаштування необхідних налаштувань безпеки, використовуючи вбудовані можливості CMS.

Цікаво те, що запит Magento в Google зустрічається набагато частіше, аніж електронна комерційна, що говорить про вплив, довіру та популярність платформи. На рисунку 1.2 зображено логотип CMS Magento.

Рисунок 1.2 - Логотип CMS Magento

1.1.3 Принцип роботи CMS системи

Принцип роботи CMS систем, що використовується для розробки веб-сайтів та мобільних додатків – це можливість створювати, редагувати та публікувати різні фрагменти цифрового вмісту (текст, зображення, відео) без написання коду, що дуже зручно для роботи великої команди контент фахівців, або навіть самого власника бізнесу, якщо мова йде про невеликий веб-сайт.

Основне завдання будь якої CMS системи – це генерувати сторінки за запитом власника задля створення чи оновлення контенту на сайті і відповідно демонструвати створенні сторінки на відповідних сторінках веб-сайту для користувачів, та відвідувачів сайту.

На рисунку 1.3 зображений принцип генерації сторінок за запитом контент менеджера. Тобто контент менеджер використовуючи модуль редагування створює чи оновлює одну зі сторінок на сайті, після чого нова інформація відправляється до бази даних та за допомогою вже модуля подання (back-end) частини системи стає доступною для користувача. Для контент менеджера та користувача процес створення та відображення настільки швидкий, що вони навіть не встигають помітити, що система виконує ще низку додаткових дій.

Як можна побачити та підсумувати принцип генерації сторінок та контенту на рисунку 1.3 CMS системи працюють на основі зв’язки, де інформація в базі даних змінюється за допомогою модуля редагування.

Модуль подання генерує сторінку із змістом при запиті на нього, на основі інформації з бази даних. Сторінки наново створюються сервером (модулем подання) при кожному запиті. На основі URL у модулі запиту відбувається визначення запитуваної порції контенту/сторінки (наприклад через параметри методу GET) [6].

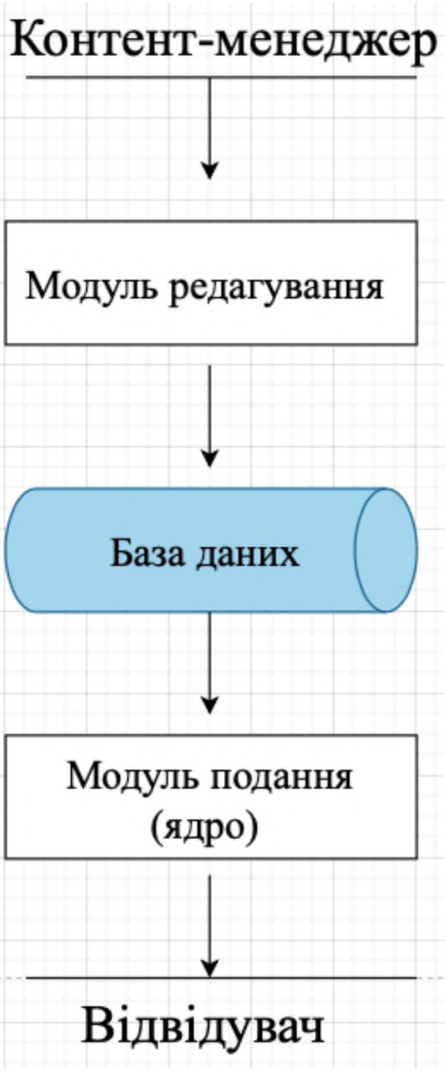


Рисунок 1.3 - Принцип генерації сторінок за запитом контент-менеджера

CMS система дозволяє створювати, редагувати та публікувати необхідний контент швидко, та без зусиль. Це найбільш актуально для інтернет магазинів, інформаційних сайтів, де контент сайту потрібно оновлювати часто і використовуючи CMS систему можна швидко навчити людину, контент менеджера, який буде займатись саме оновленням контенту сайту без програмування.

Мова йде не тільки про контент сторінки для сайту, це може будуть й каталоги товарів, коли це інтернет-магазин, різні налаштування акцій та їх відображення на сайті.

На рисунку 1.4 зображена схема роботи CMS системи.

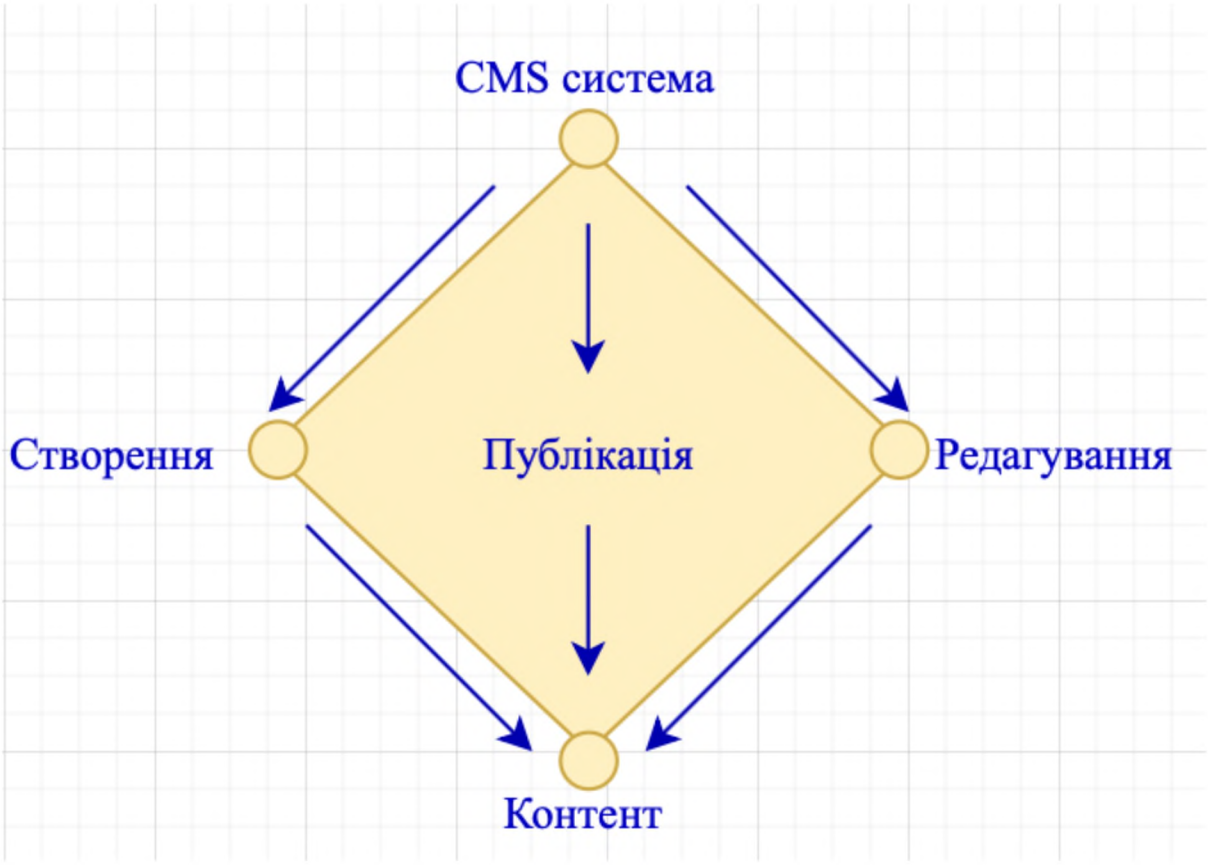


Рисунок 1.4 - Схема роботи CMS системи

Якщо коротко описати архітектуру CMS, то вона складається з середовища програмування, бази даних для зберігання даних та рівня презентації з набором шаблонів для налаштувань макета.

Головним принципом CMS є те, що вміст веб-сайту та його дизайн відокремлені.

Дизайн веб-сайту рідко модернізується, тоді як зміст слід міняти щодня або навіть годину. Тому багато CMS використовують шаблони.

Шаблон – це порожній заголовок сторінки із вбудованими компонентами дизайну веб-сайту. Прагнучи змінити вміст, слід лише заповнити цей шаблон

необхідною інформацією та зберегти зміни. Для більш детального розуміння на рисунку 1.5 представлена архітектура CMS системи.

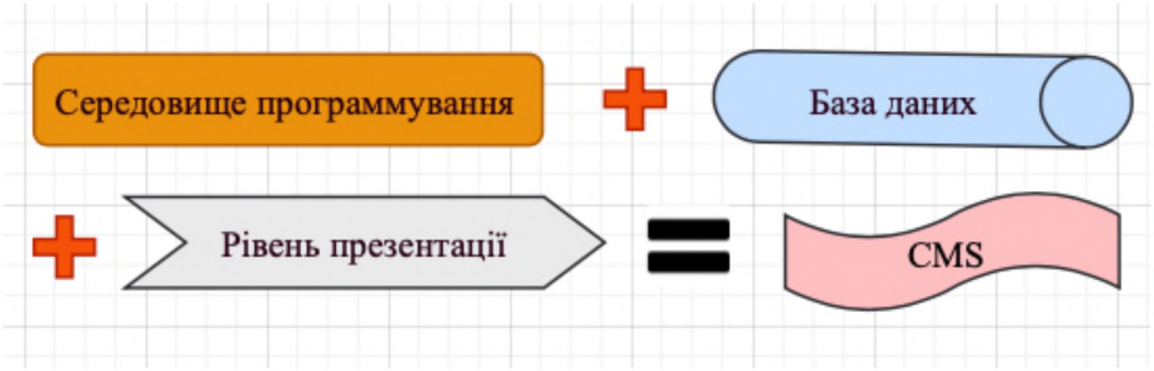


Рисунок 1.5 - Архітектура CMS системи

На більш технічному рівні система управління вмістом складається з двох великих структурних складових частин:

CDA – внутрішня частина, відповідальна за логіку коду веб-рішення та за перетворення вмісту у візуальну частину для кінцевих користувачів.

CMA – інтерфейсна частина або користувальницький інтерфейс для редакторів, де вони створюють та оновлюють вміст, не торкаючись коду.

Ці два компонента тісно пов'язані, але нетехнічні користувачі працюють на рівні інтерфейсу і не бачать бекенда, бо за це відповідають розробники та веб-майстри.

Наприклад, коли ви пишете нову статтю у своєму щоденнику або додаєте детальну інформацію на сторінку товару вашого Інтернет-магазину, ви завантажуєте текст та зображення у стандартний шаблон і натискаєте кнопку "Опублікувати". На даний момент всі дані, які ви додали до шаблону, надсилаються до бази даних, де у відповідній таблиці створюється новий рядок для зберігання цих даних.

Основною особливістю CMS є так званий принцип WYSIWYG (те, що ви бачите, це те, що ви отримуєте), який означає, що вміст, який ви бачите в процесі редагування, виглядає майже так само, як він з'явиться на веб-сторінці, коли ви його опублікуєте для користувачів [4].

CMS не вимагає встановлення додаткового програмного забезпечення, оскільки воно знаходиться безпосередньо на сервері. Ви можете отримати доступ до нього за допомогою браузера. CMS підтримує різні широко поширені браузери, такі як Internet Explorer, Mozilla Firefox, Opera тощо.

1.1.4 Переваги та недоліки CMS систем

Як і кожна система, CMS система також не ідеальна і має свої недоліки та переваги. Тому перед тим, як остаточно обрати CMS систему для свого веб-сайту, необхідно враховувати її плюси та мінуси. Знання та розуміння сильних та слабких сторін допоможе у майбутньому зекономити кошти на додаткових конфігурацій та налаштувань які можуть знадобитись через недостатньо повних функціонал чи помилки у конфігураціях.

У таблиці 1.1 проаналізовані та представлені основні переваги та недоліки CMS систем.

Таблиця 1.1 - Переваги та недоліки CMS систем

Переваги	Недоліки
Швидкий час розробки, що дозволяє швидко створити та опублікувати веб-сайт без допомоги розробників або спеціальних фахівців.	Стандартний функціонал та логіку неможливо змінити без додавання чи змінювання стандартного коду. Тобто самостійно ви можете використовувати лише стандартні функції обраної системи.
Дизайн та редагування - це два різні процеси. Навіть недосвідчені користувачі можуть створювати, додавати, редагувати та форматувати вміст веб-сайтів без особливих знань.	Створення унікального дизайну, може бути проблемою, адже зазвичай використовуються готові шаблони, які важко кастомізувати та змінювати. Тобто самостійно, ви можете лише використовувати готову тему або шаблони.
Доступна вбудована перевірка граматики, щоб уникнути помилок при оновленні чи додаванні контенту.	Система може “зламатися”, якщо адмін неправильно зробив або змінив налаштування

Продовження таблиці 1.1

Доступ базується на ролях. Кожному користувачеві CMS присвоюється певна роль (автор, редактор, співавтор тощо). Деякі користувачі можуть отримати певні права на зміну вмісту; інші можуть мати універсальний доступ. Цей підхід допомагає підтримувати безпеку доступу, оскільки він обмежений відповідно до ваших уподобань.	Великі ризики, пов'язані з безпекою. CMS платформа – це здобич для атак хакерів.
Швидкі оновлення. CMS дозволяє оновлювати ваш веб-сайт, не докладаючи особливих зусиль.	Система вимагає оновлення - а це в свою чергу вимагає часу і потребує відповідальних людей, щоб забезпечити його постійне оновлення.
SEO-зручний інтерфейс. Типова система управління вмістом включає заголовки сторінок, метадані та регульовані URL-адреси. Також для оптимізації доступні допоміжні плагіни.	SEO налаштування може викликати ряд труднощів, які не можуть вирішити просто користувачі, і може знадобитись додаткова допомога спеціалістів.
Безкоштовне завантаження. Багато основних функцій системи управління вмістом є безкоштовними і їх можна завантажити з Інтернету.	Може знадобитися додаткове фінансування та допомога розробників, якщо CMS системою користуються новачки. Або якщо система потребує додаткових кастомізацій чи зміни логіки чи функціоналу, бо функціонал неможливо змінити без додавання чи змінювання стандартного коду.

Підсумовуючи можна зробити висновок, що CMS є корисним інструментом для створення контенту веб-сайту. Недосвідчені користувачі можуть легко навчитися створювати веб-сайти за допомогою цього вдосконаленого інструменту.

Однак використання CMS пов'язане з певними плюсами і мінусами, яких не уникнути і які потрібно враховувати при роботі з CMS системою [3].

1.1.5 Статистика зломів сайтів розроблених на різних CMS системах

Кожен розробник, спеціаліст з кібербезпеки чи власник веб-сайту повинні знати про стан та цифри статистики злому веб-сайтів розроблених на різних CMS системах. Як відомо, практично кожне побудоване програмне забезпечення можна певним чином "зламати", і статистика дасть деяке розуміння, на що слід звернути увагу при розробці сайту та налаштувань безпеки та як змінюється статистика з плином часу.

Зростання кількості зломів в першу чергу пов’язане зі зростанням кількості користувачів. У таблиці 1.2 наведені статистичні дані зростання кількості користувачів с 1995 року по 2020 рік. Зараз більше 88% світового населення є інтернет користувачами, а це приголомшений відсоток.

Таблиця 1.2 - Статистика зростання інтернет користувачів

Рік	Кількість користувачів (мільйонів)	% від світового населення
1995	16	0,4
1998	147	3,6
2001	513	8,6
2004	817	12,7
2007	1319	20
2010	1971	28,8
2013	2808	39
2016	3696	49.5
2019	4422	57,3
2020	6921	88,7

Також було проведено дослідження, в якому зазначалося, що в середньому кожні 39 секунд відбувається атака в Інтернеті, а незахищені імена користувачів та паролі, які використовуються, дають зловмиснику більше шансів на успіх [7].

Хакери крадуть 75 записів щосекунди. Це середня кількість викрадених записів за секунду. Порушення, взагалі, насправді рідкісні, але коли вони трапляються, є багато записів, які викладаються всі відразу за кілька секунд або хвилин.

В середньому 300000 нових веб-сайтів взламуються щодня.

30 000 з цих сайтів зазвичай є законними сайтами малого бізнесу, які мимоволі поширюють зловмисне програмне забезпечення [7].

На рисунку 1.6 представлена статистика зломів сайтів розроблених на різних CMS системах станом на 2019.

Структура WordPress та зручність його використання роблять його першою за даними CMS для злому. Звичайно, WordPress дуже серйозно ставиться до безпеки і намагається бути насправді на крок попереду хакера. Але, деталі CVE показують інше.

За даними CVE Details, XSS (38,1%) залишається найбільш вразливою частиною в WordPress, після чого йде Якість коду (15,3%) і можливість зловмисника обійти та легко отримати інформацію (12,7%) [7].

Як можна побачити на рисунку 1.6 - CMS Magento знаходиться на другому місці і має 4,6 %, що просто кардинально відрізняється від статистики WordPress.

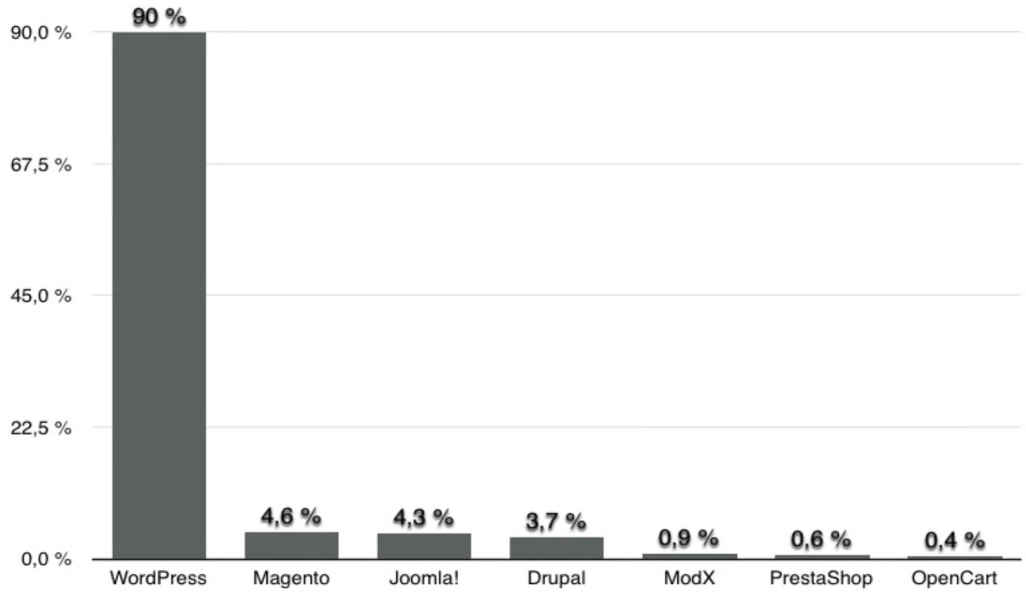


Рисунок 1. 6 - Статистика зломів сайтів розроблених на різних CMS системах

1.1.6 Аналіз нормативно-правової бази у сфері захисту інформації

Для забезпечення захисту інформації існує ціла низка напрямів забезпечення інформаційної безпеки, які направлені на зниження імовірності виникнення загрози та порушення базових властивостей інформації. Нормативно-правове забезпечення – це первинний етап при тестуванні та налаштуванні необхідного рівня безпеки для будь-якого веб-сайту.

Під час тестування веб сайту та розробки методики були прийняті до уваги такі базові нормативні документами:

- НД ТЗІ 1.1-002-99: Загальні положення з захисту інформації в комп'ютерних системах від НСД (введено в дію Наказом ДСТСЗІ СБУ від 28.04.1999 р. № 22) визначає методологічні основи (концепцію) вирішення завдань захисту інформації в комп'ютерних системах і створення нормативних і методологічних документів, регламентуючих питання щодо: визначення вимог щодо захисту комп'ютерних систем від несанкціонованого доступу; створення захищених комп'ютерних систем і засобів їх захисту від несанкціонованого доступу; оцінки захищеності комп'ютерних систем та їх придатності для вирішення завдань споживача [8];
- НД ТЗІ 1.1-003-99: Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу (введено в дію Наказом ДСТСЗІ СБУ від 28.04.1999 р. № 22) пояснює основні терміні у галузі захисту інформації [9];
- НД ТЗІ 2.5-004-99: Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу (введено в дію Наказом ДСТСЗІ СБУ від 28.04.1999 р. № 22) [10];
- НД ТЗІ 2.5-010-03 Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу (прийнято 2003-04-02) [11];
- Стандарт захисту даних платіжних карток (PCI DSS) - стандарт безпеки даних індустріальних платформних карт, розроблений Радою зі стандартної безпеці індустріальних платіжних карт (Рада стандартів безпеки платіжних карток, PCI SSC), створений міжнародними платними платформами Visa, MasterCard, American Express, JCB [12];

- ДСТУ ISO / IEC TR 20004: 2017 Інформаційні технології. Методи захисту. Уточнений аналіз вразливості програмного забезпечення відповідно до ISO / IEC 15408 та ISO / IEC 18045 (ISO / IEC TR 20004: 2015 року, IDT) [13].

1.2 Постановка задачі

З огляду на актуальність проблеми та велику кількість атак на веб-сайти розроблених за допомогою CMS систем, та недостатньої кваліфікації користувачів та власників бізнесів, можна сформувати наступні задачі для кваліфікаційної роботи:

- проаналізувати основні вразливості веб-сайті, які найчастіше стають причиною злому сайту;
- визначити принцип роботи та принцип реалізації атак за допомогою SQL-ін’єкцій для веб-сайтів розроблених на CMS Magento;
- розробити методику тестування та виявлення SQL ін'єкцій для веб-сайтів розроблених на CMS Magento;
- економічно обґрунтувати доцільність використання методики тестування та виявлення SQL ін'єкцій.

1.3 Висновок

В першому розділі кваліфікаційної роботи було розглянуто та проаналізовано:

- поняття CMS системи, її основне призначення та особливості;
- технічні особливості та різницю у характеристиках основних CMS систем;
- принцип роботи CMS системи та розглянута та досліджена детальна архітектура;
- були виділені найпоширеніші особливості та можливості хорошої та якісної системи управління вмістом;
- представлена статистика злому сайтів розроблених за допомогою різних CMS систем;
- розглянуто нормативно-правові акти в області забезпечення та впровадження необхідного рівня на налаштувань для веб-сайту;
- виконано поставку задачі для кваліфікаційної роботи.

РОЗДІЛ 2. СПЕЦІАЛЬНА ЧАСТИНА

2.1 Основні вразливості сайту

Згідно зі стандартом OWASP було виділено та розглянуто 10 основних вразливості сайту, які найчастіше зустрічаються при аналізі сайту та стають причинами злому хакерів [2]. Вразливості однаково актуальні для самописних систем так і для веб-сайті розроблених за допомогою CMS систем.

1. Ін'єкція.

Вразливість ін’єкції, така як SQL, NoSQL, OS та ін’єкція LDAP, що виникають, коли ненадійні дані надсилаються інтерпретатору як частина команди або запиту. Ворожі дані зловмисника можуть змусити систему виконати ненавмисні команди або отримати доступ до даних без належного на те дозволу.

2. Порушена автентифікація.

Функції додатків, пов’язані з автентифікацією та управлінням сеансами, часто реалізовані неправильно, що дозволяє зловмисникам скомпрометувати паролі, ключі або маркери сеансу або використовувати інші недоліки реалізації, щоб тимчасово або назавжди отримати ідентифікаційні дані інших користувачів.

3. Експозиція чутливих даних.

Багато веб-додатків та API не захищають належним чином конфіденційні дані, такі як фінанси, охорона здоров'я та ідентифікація. Зловмисники можуть викрасти або модифікувати такі слабо захищені дані для здійснення шахрайства з кредитною карткою, викрадення особистих даних або інших злочинів. Конфіденційні дані можуть бути скомпрометовані без додаткового захисту, наприклад, шифрування в спокої або під час передачі, тому вони вимагають особливих заходів обережності при обміні ними з браузером.

4. Зовнішні організації XML (XXE).

Багато старих або погано налаштованих процесорів XML оцінюють посилання на зовнішні сутності в документах XML. Зовнішні сутності можуть використовуватися для розкриття внутрішніх файлів за допомогою обробника URI

файлів, внутрішніх спільних файлів, внутрішнього сканування портів, віддаленого виконання коду та атак відмови в обслуговуванні.

5. Порушений контроль доступу.

Обмеження щодо дозволених користувачів, що мають автентифікацію, часто не застосовуються належним чином. Зловмисники можуть використовувати ці недоліки для доступу до несанкціонованих функціональних можливостей та / або даних, таких як доступ до облікових записів інших користувачів, перегляд конфіденційних файлів, зміна даних інших користувачів, зміна прав доступу тощо.

6. Неправильна конфігурація безпеки.

Неправильна конфігурація безпеки - це найчастіша проблема. Зазвичай це результат небезпечних конфігурацій за замовчуванням, неповних або спеціальних конфігурацій, відкритого хмарного сховища, неправильно налаштованих заголовків HTTP та детальних повідомлень про помилки, що містять конфіденційну інформацію. Не тільки всі операційні системи, фреймворки, бібліотеки та програми повинні бути надійно налаштовані, але вони повинні бути вчасно виправлені / оновлені.

7. Міжсайтовий скриптинг XSS.

Недоліки XSS трапляються, коли програма включає ненадійні дані на новій веб-сторінці без належної перевірки чи екранування, або оновлює наявну веб-сторінку даними, наданими користувачем, за допомогою API браузера, який може створювати HTML або JavaScript. XSS дозволяє зловмисникам виконувати сценарії в браузері жертви, які можуть викрадати сеанси користувачів, псувати веб-сайти або перенаправляти користувача на шкідливі сайти.

8. Небезпечна десеріалізація.

Небезпечна десеріалізація часто призводить до віддаленого виконання коду. Навіть якщо недоліки десеріалізації не призводять до віддаленого виконання коду, їх можна використовувати для виконання атак, включаючи атаки повторного відтворення, атаки інжекції та атаки ескалації привілеїв.

9. Використання компонентів із відомими вразливостями.

Компоненти, такі як бібліотеки, фреймворки та інші програмні модулі, працюють з тими самими привілеями, що і програма. Якщо експлуатувати вразливий компонент, така атака може сприяти серйозній втраті даних або захопленню сервера. Програми та API, що використовують компоненти з відомими вразливими місцями, можуть підірвати захист додатків та активувати різні атаки та впливи.

10. Недостатня реєстрація та моніторинг.

Недостатня реєстрація та моніторинг у поєднанні з відсутністю або неефективною інтеграцією з реагуванням на інциденти дозволяють зловмисникам продовжувати атакувати системи, підтримувати стійкість, переходити до більшої кількості систем і підробляти, витягувати або знищувати дані. Більшість досліджень порушень показують, що час на виявлення порушення становить понад 200 днів, як правило, виявляються зовнішніми сторонами, а не внутрішніми процесами чи моніторингом.

На рисунку 2.7 зображено відношення різних вразливостей та їх частота зустрічання.

Як видно з діаграмами зображеної на рисунку 2.7, SQL ін'єкція на даний момент є не тільки широко поширеною вразливістю, а ще й однією з найнебезпечніших за версією OWASP Top 10 Application Security Risks, тому саме на ній зроблений акцент у роботі.

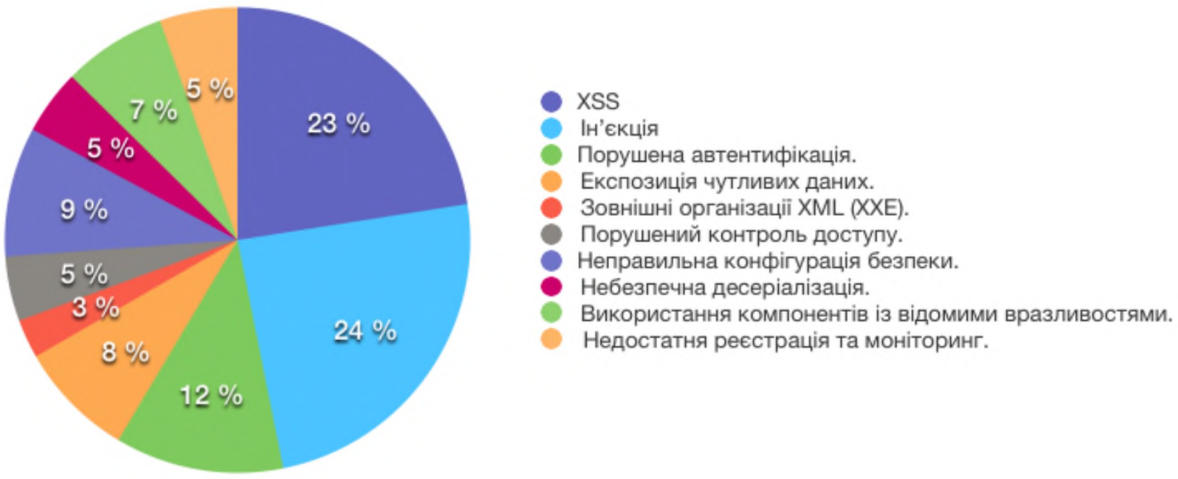


Рисунок 2.7 - Відношення різних вразливостей та їх частота зустрічання

2.2 Основні дані про SQL ін’єкції

Перші відомості про SQL-ін'єкції з'явилися у відкритому доступі в статті «NT Web Technology Vulnerabilities», опублікованій ще в 1998 році [18].

З 2000 року дану техніку почали широко застосовувати кіберзлочинці для злому інтернет-сайтів. На сьогоднішній день існує велика кількість технік експлуатації SQL-ін'єкцій для різних СУБД і додатків на їх основі.

Ін’єкція SQL – це вразливість сайту, яка дозволяє зловмисникам втручатися в запити, які сайт робить до своєї бази даних. Як правило, це дозволяє зловмиснику переглядати дані, які вони зазвичай не можуть отримати. В першу чергу це може включати дані, що належать іншим користувачам, або будь-які інші дані, до яких сама система може отримати доступ. У багатьох випадках зловмисник може змінювати або видаляти ці дані, спричиняючи постійні зміни у вмісті або поведінці сайту.

У деяких ситуаціях зловмисник може посилити атаку введення SQL для компрометації базового сервера або іншої внутрішньої інфраструктури та здійснити атаку відмови в обслуговуванні.

SQL-ін’єкція використана для веб-сайту розробленого за допомогою CMS Magento може:

- Прочитати вміст бази даних;
- Маніпулювати базою даних та може змінити контент магазину;
- Видалити всю базу даних;
- Викрасти дані кредитних картки;
- Видавати облікові дані адміністратора. Що може призвести до подальших атак [19].

Існує широкий спектр вразливостей, атак та методів введення SQL, які виникають у різних ситуаціях, але все ж SQL атаки поділяються на декілька типів.

Найпопулярніший метод, який виконують зловмисники - це введення SQL на основі об’єднання. Ця техніка введення дозволяє кіберзлочинам отримувати дані з бази даних, розширюючи результати з вихідного запиту. Метод використовує

оператор UNION SQL для інтеграції двох операторів SELECT в один результат, а потім повертає їх як частину відповіді.

Більш складною у виконанні, ніж інші різновиди ін'єкцій, яку зловмисники виконують – це сліпі ін'єкції SQL, коли від цілі надходять загальні повідомлення про помилки. Сліпі ін'єкції SQL відрізняються від звичайних ін'єкцій SQL методом отримання інформації з бази даних. У цій техніці хакери запитують у базі даних справжні чи помилкові питання, потім визначають правильну відповідь на основі відповіді, а також часу, необхідного для отримання відповіді сервера при використанні його з атаками.

Тип атаки, яка замінює логіку та умови запиту на власні, або як її ще називають Логічна ін’єкція SQL або Булева. Цей тип атаки зазвичай використовується в запитах дозволів або автентифікації, коли хакери обманюють базу даних, вважаючи, що вони мають підвищені дозволи облікових даних.

Під час введення SQL ін’єкції на основі помилок зловмисники використовують помилки бази даних із веб-сторінки або програми, які були спричинені несанітованими введеннями.

Під час атаки цей метод використовує повідомлення про помилки для повернення повних результатів запиту та розкриття конфіденційної інформації з бази даних. Цей метод також можна використовувати для визначення того, чи веб-сайт або веб-програма є вразливим, та отримання додаткової інформації для реструктуризації зловмисних запитів.

Під час звичайної ін'єкції SQL хакери можуть просто читати текст у міру його повернення. Однак, коли зловмисники не можуть отримати інформацію із сервера баз даних, вони часто застосовують ін'єкції SQL на основі часу для досягнення своїх результатів. Це працює за допомогою операцій, які виконуються довго – часто декілька сотень секунд.

Ін'єкції SQL на основі часу зазвичай використовуються для визначення наявності вразливостей у веб-програмі чи на веб-сайті, а також у поєднанні з логічними методами під час сліпих ін'єкцій SQL.

Кожен зловмисник обирає тип, в залежності від кінцевої цілі атаки. Інколи кіберзлочинці можуть використовувати декілька типів, для того аби досягти більших результатів, та вдало реалізувати атаку.

Розробляючи методику тестування необхідно врахувати кожен з наведених типів, для того аби своєчасно виявити та не допустити реалізації жодної з SQL-ін'єкції.

На рисунку 2.8 зображені основні типи SQL-ін'єкцій.

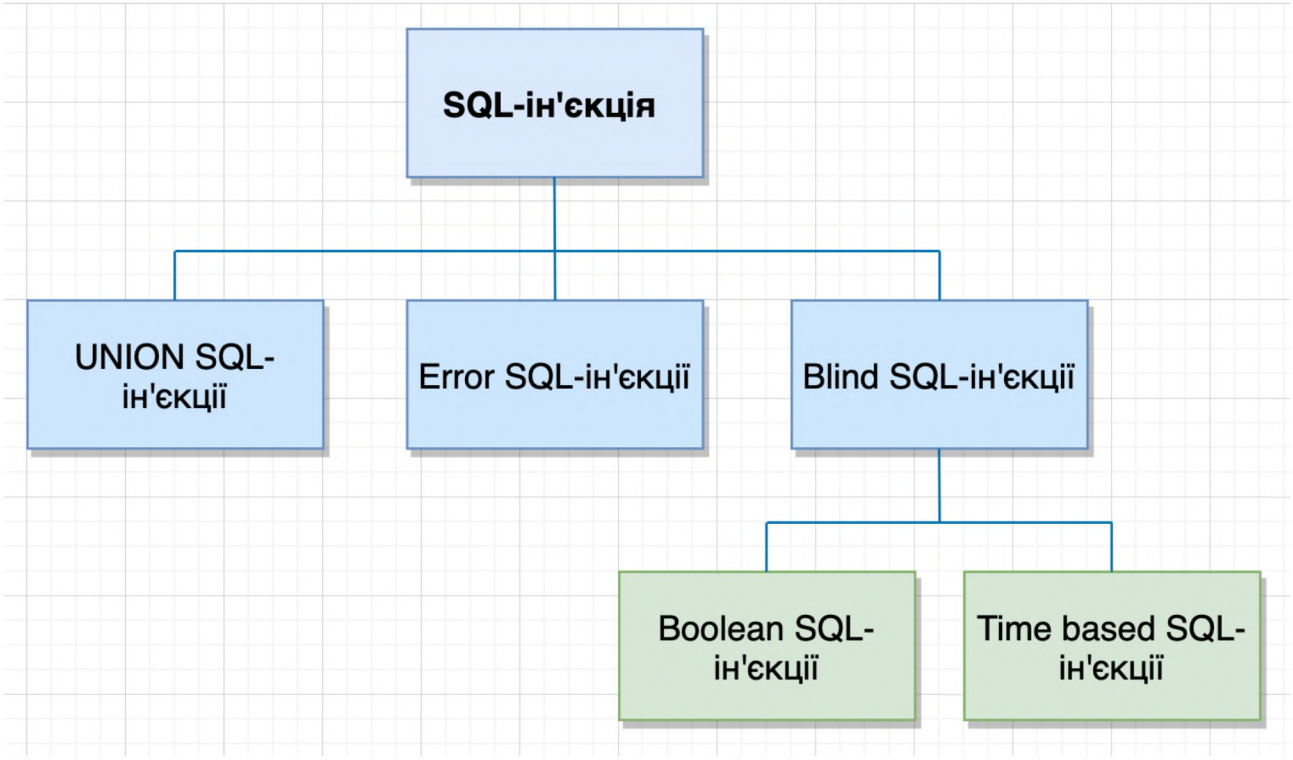


Рисунок 2.8 - Основні типи SQL-ін'єкцій

2.3 Принцип роботи SQL ін'єкції

Зазвичай, робота системи складається з трьох компонентів: користувача, сервера сайту та серверу управління базами даних. На рисунку 2.9 зображено процес взаємодії усіх трьох компонентів у разі коректної роботи системи.

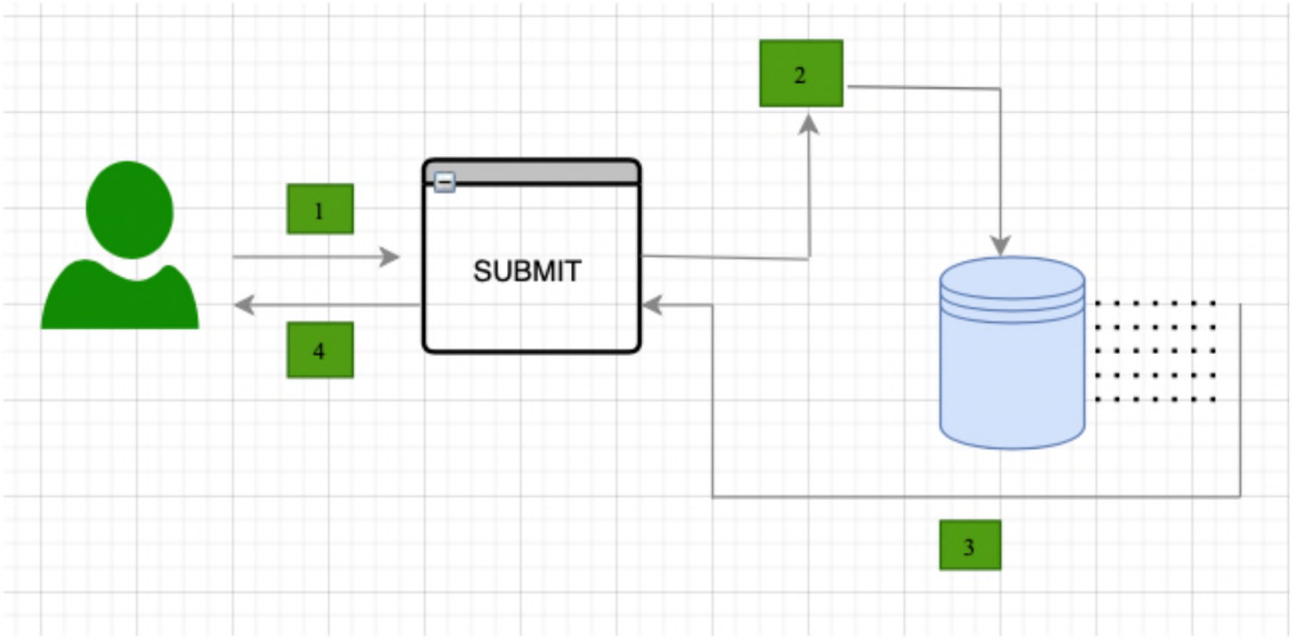


Рисунок 2.9 - Коректний процес роботи системи

Тож нормальна поведінка системи виглядає так (номер пункту відповідає та описує, що відбувається у системі, згідно з рисунком 2.9):

1. Користувач відправляє запрос http: // localhost: 8080 / catalog / show.jsp? Name = Product 1 на сервер додатків. Як значення строкового параметра name передається значення Product 1.

2. Сервер додатків динамічно формує SQL-запит: SELECT * FROM products WHERE name LIKE 'Products 1%' ORDER BY name.

3. З таблиці products витягуются дані про товари, найменування яких починається з Product 1, результат впорядковується за найменуванням товару. Дані передаються на сервер додатків.

4. Сервер додатків формує результат, який відображається користувачеві.

Тепер розглянемо, як буде працюватиме система, якщо зловмисник спробує реалізувати SQL ін'єкцію.

Приклад реалізації SQL ін'єкції, розраженний на рисунку 2.10:

1. Якщо ж в якості параметра передати, наприклад, http: // localhost: 8080 / products?category=Gifts'+OR+1=1--,

2. Сервер додатків динамічно сформує SQL - запит SELECT * FROM products WHERE category = "Gifts" or 1=1 --1' AND released = 1.

3. Запит поверне всі товари, що є в системі. Зловмисник отримує інформацію про всі товари в системі, незалежно від того доступні вони зараз на сайті чи ні.

Так само це може спрацювати і з паролями та даними користувачів та будь-якою інформацією яку хоче отримати зловмисник.

Наведений вище приклад показав, як користувач, модифікуючі SQL запит передає строковий параметр, що змусив базу даних виконувати додаткові команди.

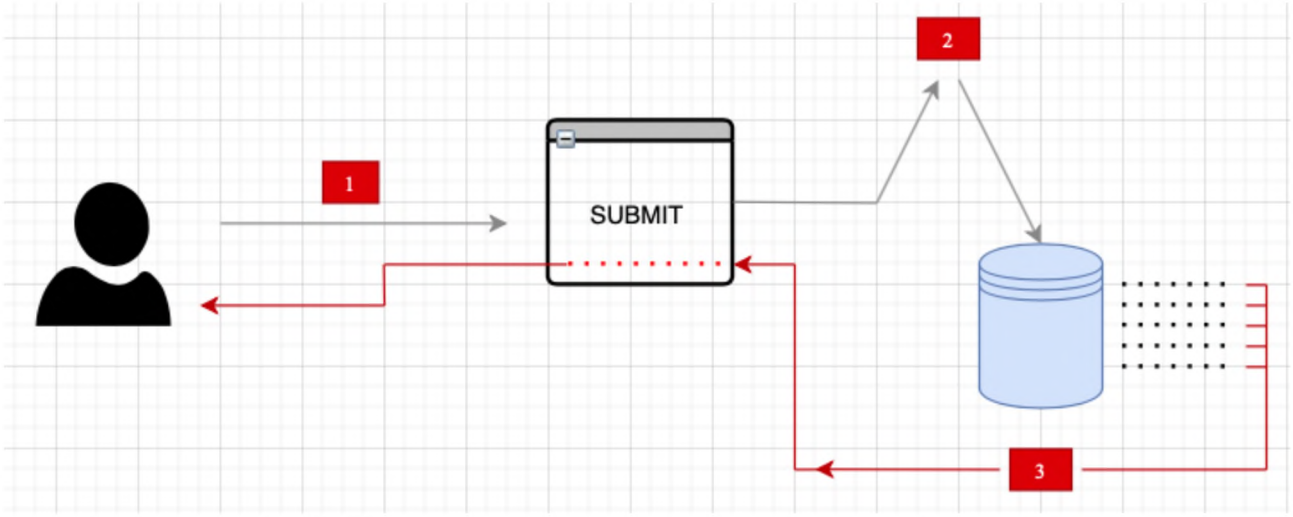


Рисунок 2.10 - Приклад реалізації SQL ін’єкції

2.4 Методика тестування та виявлення SQL ін’єкції

SQL-ін’єкції, як відомо, досить важко виявити. На відміну від міжсайтових сценаріїв, віддаленого введення коду та інших типів атак. Це пов’язано в першу чергу з тим, що ін’єкції SQL - це уразливості, які не залишають слідів на сервері. Натомість експлойт виконує справжні запити в базі даних. Як результат, більшість атак виявляються, коли зловмисник використовує вразливість для здійснення шкідливих дій або отримання адміністративного доступу [21].

Вживаючи запобіжні заходи та активно контролюючи вашу базу даних та її запити, ви можете визначити, чи здійснює зловмисник шкідливі ін’єкції на вашому веб-сайті.

Запропонована методика дозволить протестувати та виявити SQL ін’єкції та не допустити втрати конфіденційної інформації та успішної реалізації загрози.

Методика тестування та виявлення SQL ін'єкцій представляє собою аналіз бази даних, яка використовується для веб-сайту розробленому за допомогою CMS Magento; набір сценаріїв типових атак на сайт, які необхідно протестувати з урахуванням особливостей системи. У запропонованій методиці буде описано крок за кроком необхідні кроки, що повинні бути виконанні для успішного тестування та виявлення SQL-ін'єкції.

2.4.1 Оцінка досліджуваного сайту

Magento - це популярна платформа електронної комерції з відкритим кодом, в якій на даний момент працює понад 220 000 магазинів. Це робить її привабливою мішенню для хакерів.

Протягом останніх кількох років хакери використовували численні вразливості для компрометації веб-сайтів Magento та розміщення шкідливих скриптів, які викрадають дані про оплату на сторінках оплати. Тому знання про те, як захистити свій сайт просто необхідні для безпечної роботи бізнесу.

Будучи системою з відкритим кодом, Magento використовує MySql або MariaDB для зберігання та управління даними.

Розробники CMS Magento дуже уважно ставляться до питання безпеки веб-сайтів і систематично випускають оновлення, так званні Патчі безпеки, які захищають сайт від нових типів атак та вразливостей. Дуже важливо оновлювати свою систему саме цими Патчами безпеки задля підвищення рівня безпеки сайту. Але, на жаль, Патчі безпеки не можуть захистити додатковий функціонал або сторонні модулі, які були встановленні. Саме на ці модулі та кастомний функціонал спрямовують свою увагу кіберзлочинців.

Розглянемо архітектуру системи та шлях користувача до бази даних.

Користувач заходить на сайт – вводить значення у поля вводу – натискає на кнопку для застосування запиту – система перевіряє правильність введених даних і відправляє запит на сервер – сервер формує SQL запит і передає його у базу даних для отримання результатів.

Шлях користувача зображений на рисунку 2.11.

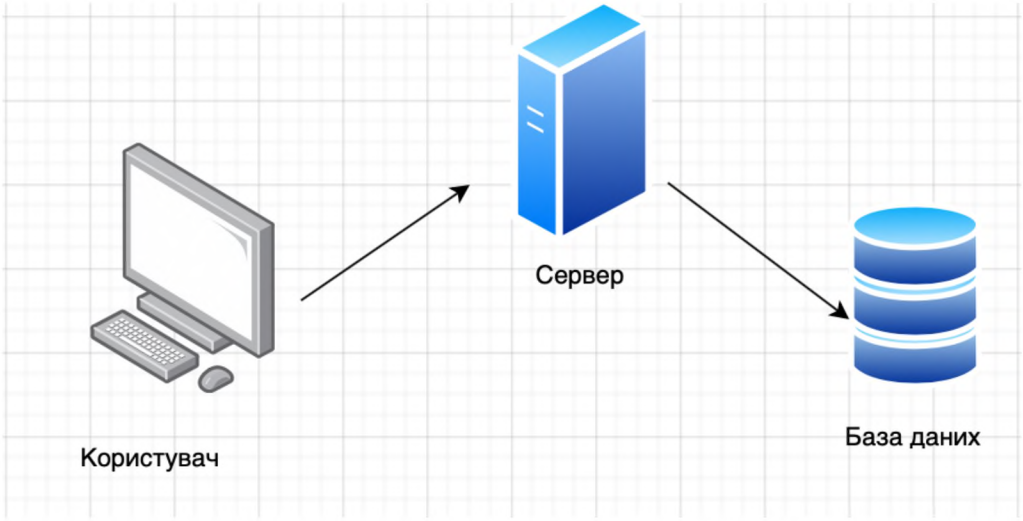


Рисунок 2.11 - Шлях користувача до бази даних

Для того аби протестувати сайт та виявити вразливість був обраний тестовий інтернет-магазин розроблений за допомогою CMS Magento.

Посилання на сайт : https://m233.dev-dinarys.com/

Почнемо тестування з оцінки досліджуваного сайту. На цьому етапі виявимо особливості сайту, мову програмування, точну версію CMS системи, що використовується і т.д.

Для цього добре підходить інструмент WhatWeb [16]. Ця утиліта надає детальну інформацію про CMS і використаних в ній веб-інструменти.

Версія та технологія програмування – це перший крок процесу збору інформації. Знання версії та типу запущеного веб-сервера дозволяє при тестуванні визначати слабкі місця системи та відповідні експлоїти.

Whatweb може ідентифікувати необхідну інформацію про веб-сайт, таку як, наприклад:

- Платформа
- Платформа CMS
- Тип сценарію
- Google Analytics
- Платформа веб-сервера
- IP-адреса, країна

За допомогою Whatweb можна провести як пасивне сканування, так і агресивне тестування. Пасивне сканування просто витягує дані із заголовків HTTP,

що імітують звичайне відвідування. Агресивні параметри поглиблюються завдяки рекурсії та різним типам запитів, а також ідентифікують усі технології, як сканер вразливостей.

Для ідентифікування інформації про CMS систему та використаних у ній веб-інструментів було проведено пасивне сканування веб-сайту: https://m233.dev-dinarys.com/ за допомогою утиліти Whatweb. На рисунку 2.12 представлені результати дослідження сайту.

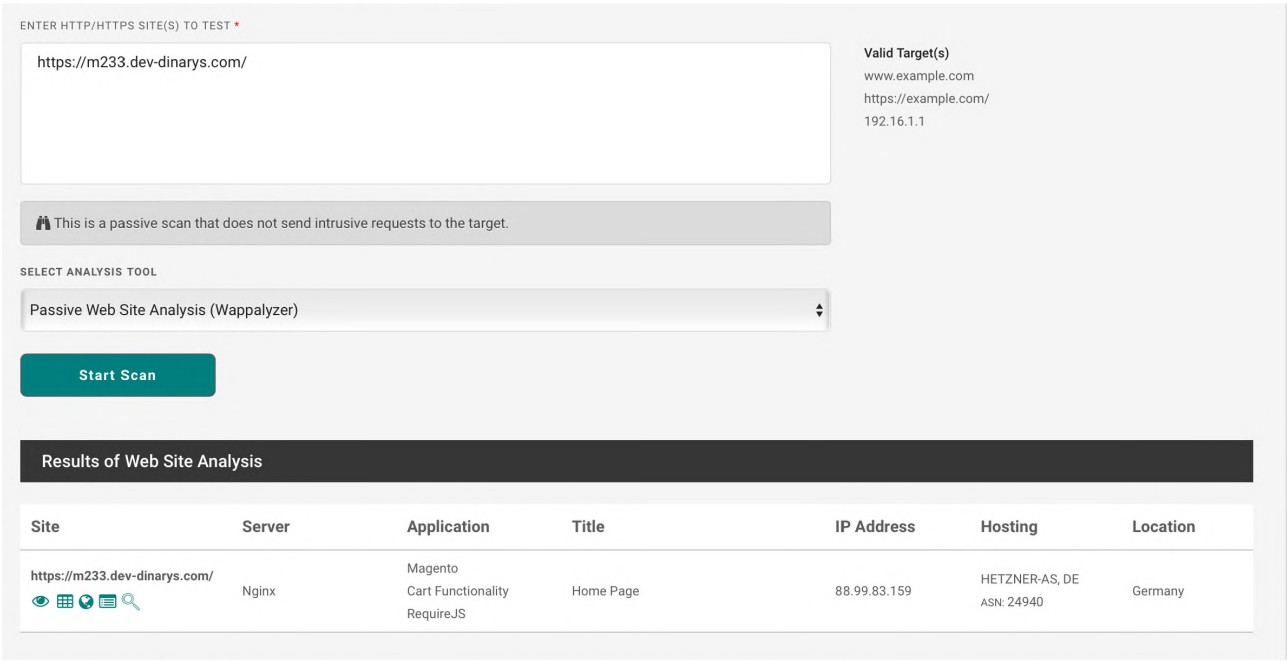


Рисунок 2.12 - Результат сканування сайту на тип CMS і використаних в ній веб-інструментів

Як видно з результатів тесту, сайт розроблений за допомогою CMS Magento, можна дізнатися інформацію про сервер, IP адресу, хостинг та локацію розташування домену.

За допомогою Whatweb нам вдалося отримати дані, які система витягує із заголовків HTTP, що імітують звичайне відвідування, тому для отримання ще більш детальної інформації про сайт, застосуємо ще одну утиліту для сканування сайтів SUCURI [17].

За допомогою цього застосунку можна отримати додаткову інформацію про сайт та рівень безпеки сайту. Цей застосунок добре підходить саме для CMS Magento і може перевірити актуальність патчів безпеки та необхідність оновлення

системи. Також SUCURI дає рекомендації для підвищення рівня безпеки конкретного веб-сайту.

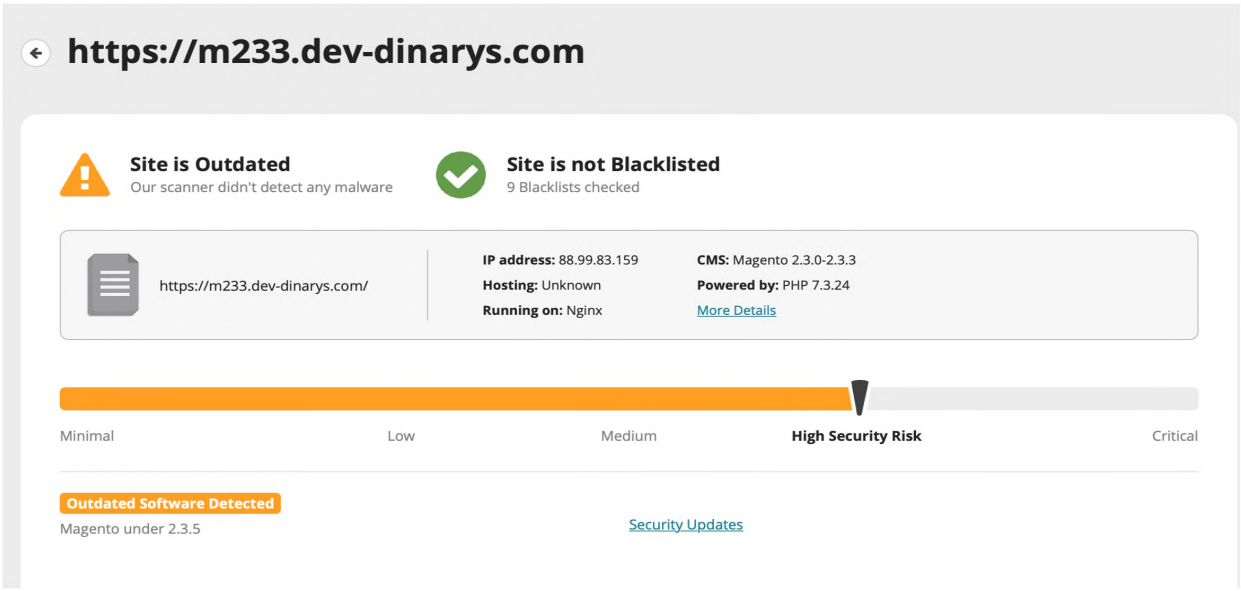


Рисунок 2.13 - Інформація про веб-сайт

На рисунках 2.13 та 2.14 представлена розширена інформація про досліджуваний сайт після сканування, використовуючи утиліту SUCURI Site Check. Повний звіт та результати тестування представленні у Додатку Г та Ґ.

За результатами дослідження виявлено, що сайт має високий рівень небезпеки.

Точна версія CMS системи, що використовується – це Magento 2.3. IP адреса – 88.99.83.159, Мова програмування, що використовується – PHP 7.3.24.

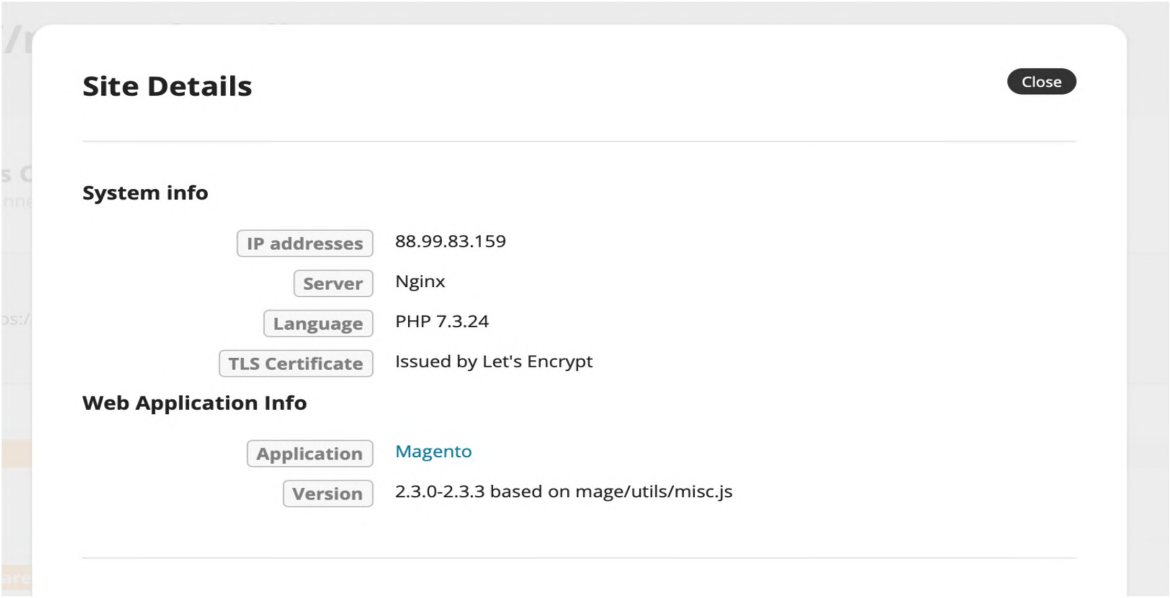


Рисунок 2.14 - Детальна інформація про сайт

То ж за допомогою 2-х застосунків WhatWeb та SUCURI Site Check нам вдалося провести оцінку особливостей сайту, який необхідно протестувати.

Як підсумок після проведення першого етапу тестування ви повинні отримати інформацію про сайт, а саме:

- CMS система
- Версія CMS системи
- Мова програмування
- Веб-інструменти
- IP-адреса
- Хостинг
- Сервер
- Розташування домену
- Список додаткових модулів та застосунків, що використовуються в системі
- Присутній чи кастомний код на сайті
- Чи були зміни у логіці CMS Magento

Отримавши всю зазначену вище інформацію, можна переходити до другого етапу тестування - “Створення базової середи тестування”.

2.4.2 Створення базової середи для тестування

На цьому етапі необхідно виконати розгортку компонентів тестової середи та створити дерево файлів та каталогів. Іншими словами визначити внутрішню структуру сайту та досліджуваної системи.

На етапі створення базової середи для програмування необхідно визначитись зі структурою бази даних досліджуваного сайту для того, аби далі правильно та чітко сформувати тестові сценарії та якомога повноцінно протестувати досліджуваний сайт.

Основна мета тестової середи – дати можливість тестувальнику у повній мірі протестувати необхідний код за допомогою автоматичних перевірок чи мануальних методів при цьому тестування не повинно заважати, або зупиняти роботу сайту.

В свою чергу структура сайту – це шляхи отримання користувачем інформації від сайту. Знання структури сайту допоможе при тестуванні, а також стане у нагоді при пошуку критичних Web-сторінок для продовження тестування.

На рисунку 2.15 представлено дерево файлів та каталогів досліджуваного сайту.

Як результат після виконання другого етапу тестування у вас повинна бути готова та налаштована тестова середа для проведення тестування, а також досліджено і сформовано дерево файлів та каталогів досліджуваного сайту.

Дерево файлів та каталогів на наступних етапах буде необхідно для правильного формування тестових сценаріїв та пошуку критичних Web-сторінок.

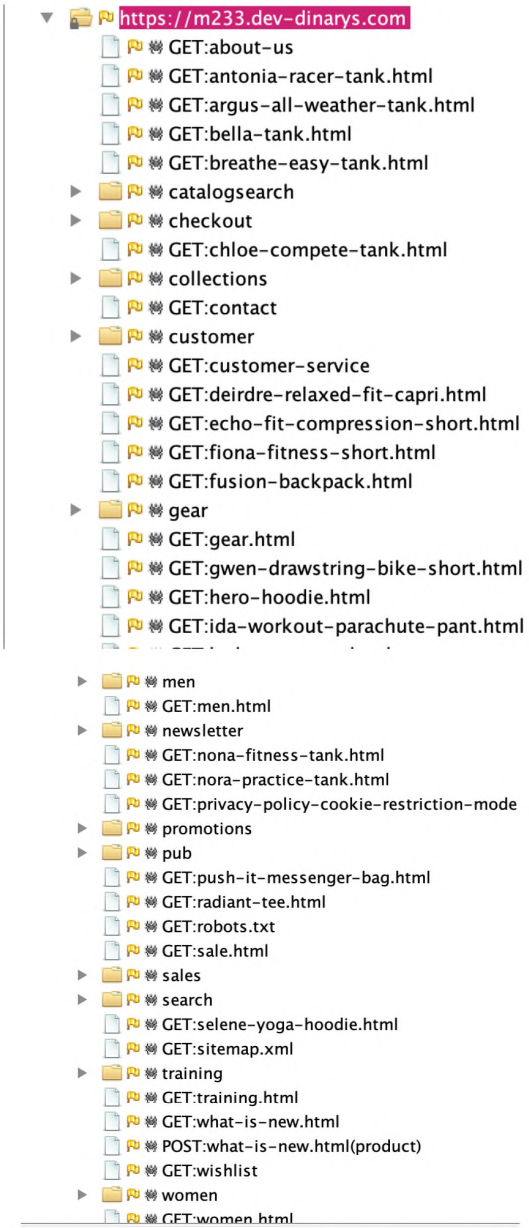


Рисунок 2.15 - Дерево файлів та каталогів досліджуваного сайт

2.4.3 Пошук критичних Web-сторінок

Існує декілька вразливих місць, через які можлива реалізація SQL ін'єкції. Такі місця можна класифікувати наступним чином:

- параметри в URL;
- поля input HTML-форми;
- hidden поля HTML - форми;
- cookies.

SQL ін'єкція, що реалізується через параметри URL - такий вид ін'єкції можливий, коли параметри передаються на сервер методом GET.

SQL ін'єкція, що реалізується через поля input HTML форми - цей вид ін'єкції можливий коли хакер намагається ввести свій SQL-запит, використовуючи поля input на Web-сторінці.

SQL ін'єкція, що реалізується через hidden поля input HTML форми - такий вид SQL атаки можливий, коли хакер намагається змінити hidden поля у WEB формі, а потім передати цю форму на сервер для виконання команди.

SQL ін'єкція, що реалізується через cookies - цей вид ін'єкції можливий, коли хакер намагається змінити значення cookies, які зберігаються на машині користувача, а потім відкрити WEB-сторінку, що використовує ці Cookies. Якщо, розробник не перевіряє значення які були отримані через cookies, то велика вірогідність того, що атака буде успішною. Дані, що були отримані з Cookies, формують SQL запит, то ж таким чином може бути сформований неочікуваний SQL запит, що стане причиною успішної реалізації SQL-ін'єкції.

На третьому етапі тестування необхідно знайти, проаналізувати та виявити Web-сторінки, які можуть мати параметри критичні по відношенню до SQL-ін'єкції.

В першу чергу це сторінки, які запитують у вас дані. Це можуть бути сторінки пошуку, різних обговорень, відгуків і тд.

Інколи html сторінки використовують метод POST, щоб відправити команди з іншої Web-сторінки. В такому випадку, ви не побачите параметри в URL, але ви можете шукати тег "FORM" в коді HTML сторінки.

Ви можете знайти, щось на кшталт такого:

<FORM action=Search/search.asp method=post>

<input type=hidden name=A value=C>

</FORM>

Всі параметри між <FORM> та </FORM> можуть бути потенційно вразливі до введення SQL кода.

Також спробуйте знайти сторінки типу ASP, JSP, CGI або PHP Web сторінки.

Також вразливі до введення SQL-ін'єкцій можуть бути сторінки, що використовують параметри типу:

https://m233.dev-dinarys.com/?ID=12345

Був визначений список Web-сторінок, які можна вважати критичними та через які можлива реалізація SQL ін'єкція, для досліджуваного сайту.

Для успішного пошуку та аналізу використовувались правила та рекомендації, що були зазначені у третьому етапі тестування, а саме пошук критичних Web-сторінок.

Список критичний Web-сторінок для досліджуваного веб-сайту:

- https://m233.dev-dinarys.com/customer/account/create/
- https://m233.dev-dinarys.com/customer/account/login/referer/aHR0cHM6Ly9tMjMzLmRldi1kaW5hcnlzLmNvbS9jdXN0b21lci9hY2NvdW50L2NyZWF0ZS8%2C/
- https://m233.dev-dinarys.com/push-it-messenger-bag./reviews/
- https://m233.dev-dinarys.com/sales/guest/form/
- https://m233.dev-dinarys.com/contact/
- https://m233.dev-dinarys.com/catalogsearch/advanced/

Як результат проведення третього етапу тестування у вас повинен бути сформований список потенційно критичних Web-сторінок, через які можлива реалізація SQL-ін'єкції.

2.4.4 Визначення критичних значень для параметрів

Четвертий етап тестування представляє собою аналіз та визначення очікуваних параметрів та їх типів для кожної з виявленої сторінки на попередньому етапі тестування. На цьому етапі вам необхідно проаналізувати та визначити параметри, які система очікує та отримує у разі коректної роботи системи. Визначенні параметри стануть у нагоді при формування тестових сценаріїв для проведення тестування.

За допомогою визначення критичних параметрів та їх типів буде необхідно сформувати тестові сценарії, а саме комбінацію очікуваних параметрів системи з неочікуваними.

Наведемо приклад визначення очікуваних параметрів для однієї з виявлених критичних Web-сторінок.

- https://m233.dev-dinarys.com/customer/account/create/

Очікуванні параметри в кожному із поля input в HTML-форми досліджуваної сторінки:

<input type="text" id="firstname" name="firstname" value="" title="First Name" class="input-text required-entry" data-validate="{required:true}" autocomplete="off" aria-required="true">

<input type="text" id="lastname" name="lastname" value="" title="Last Name" class="input-text required-entry" data-validate="{required:true}" autocomplete="off" aria-required="true">

<input type="email" name="email" autocomplete="email" id="email_address" value="" title="Email" class="input-text" data-validate="{required:true, 'validate-email':true}" aria-required="true">

<input type="password" name="password" id="password" title="Password" class="input-text valid" data-password-min-length="8" data-password-min-character-sets="3" data-validate="{required:true, 'validate-customer-password':true}" autocomplete="off" aria-required="true">

Аналогічно необхідно визначити очікувані параметри для кожної з досліджуваної сторінки.

Як результат після аналізу та визначення у вас буде сформований список очікуваних параметрів для кожної зі сторінки, що була визначена як критичною для досліджуваного сайту. Визначенні параметри будуть необхідні для правильного формування тестових сценаріїв та проведення безпосереднього тестування.

2.4.5 Підготовка до запуску тестування

Наступний, п’ятий етап тестування включає в себе підготовку до запуск самого тестування.

Тестування сайту для виявлення SQL ін’єкцій буде включати в собі тестуючий додаток (що імітує дії користувача) та створює і запускає тести та сайт, що тестується.

Далі, в схемі повинен бути налаштований proxy сервер, що імітує базу даних. В такому випадку, взаємодія сайту с базою даних буде відбуватись через цей компонент і дозволить провести більш глибоке та якісне тестування системи.

Якщо ж немає змоги встановити та налаштувати proxy сервер, то зв’язок з базою даних буде прямий, а це може впливати та аналіз відповідей від системи. Бо, наявність у схемі тестування proxy сервера дозволяє підвищити точність тестування. Proxy сервер,отримує SQL запит від сайту, оцінює його та передає серверу бази даних, такий спосіб є більш точним та інформативним при проведенні тестування та виявлення вразливостей.

На рисунку 2.16 зображена схема тестування, що повинна бути створена для правильного запуску тестування.

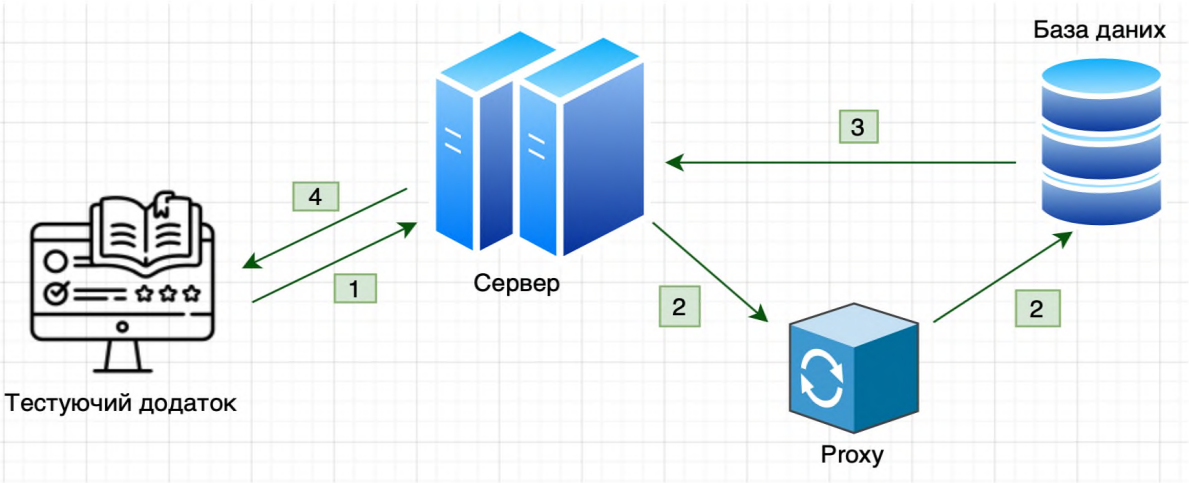


Рисунок 2.17 - Схема тестування

Умовні позначення:

1 - Тестовий запит

2 - SQL запит

3 - Відповідь бази даних

4 - Відповідь сайту / відображення результатів

Закінчуючи п'ятий етап тестування у вас повинна бути створена на налаштована правильна схема тестування, що складається з тестуючого додатку, серверу, Proxy серверу та бази даних згідно рисунку 2.17.

2.4.6 Методологія тестування/Запуск тесту

Наступним етапом буде запуск тесту згідно з тестових сценаріїв. У методиці запропонований шаблон сценарію, аби тестувальник у разі необхідності міг створити додатково необхідну кількість тестових сценаріїв для повної перевірки згідно з виявленими вразливими Web-сторінками досліджуваного сайту та 5 тестових сценаріїв, які можна використовувати для тестування та перевірки основних критичних параметрів.

Розглянемо шаблон сценарію, який буде використовуватися для тестування. Шаблон зручно використовувати для опису елементів, що повторюються для визначення тестових сценаріїв.

Вхідні дані у кожному з тестових сценаріїв будуть унікальні в залежності від типу тесту.

Для того, аби перевірити можливість реалізації SQL ін'єкції буде виконуватися додавання до очікуваного сайтом та базою даних запиту, ті символи та параметри які не очікуються і не повинні там бути присутні.

Саме тестування буде складатися з оцінки очікуваного значення параметри, потім безпосередньо додавання нових символів, що призводять до SQL ін'єкцій та оцінка отриманих результатів та відповідей від сайту.

Набір символів, що використовується для тестування унікальний для кожного з тестового сценарію.

Результати отриманих тестів визначаються з аналізу параметру, що був відправлений на сайт та параметру, що надійшов з сайту на proxy. Якщо на proxy

прийшов запит, який має у собі додаткові символи - тест можно вважати позитивним [20].

На рисунку 2.18 зображено шаблон тестування, що коротко і чітко описує етапи, які необхідно виконати для кожного з тестового сценарію.

Використовуючи, шаблон сценарію ви зможете правильно та повноцінно перевірити досліджуваний сайт.

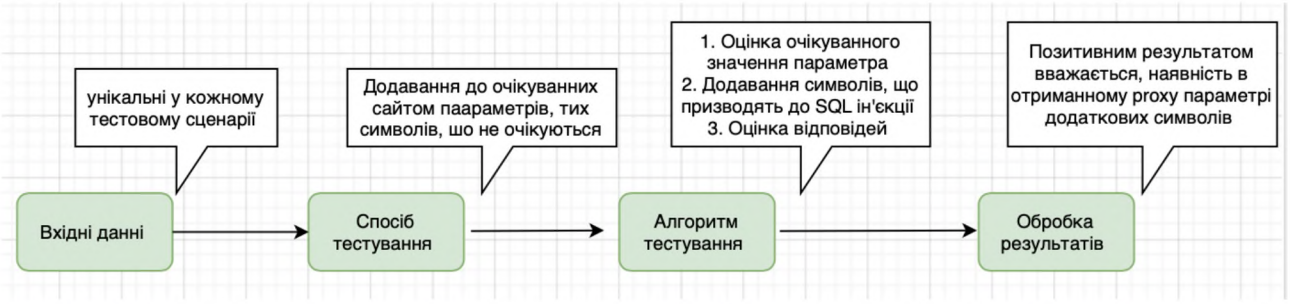


Рис 2.18 - Шаблон тестування

У методиці розроблено та запропоновано 5 тестових сценарій, кожен з яких призначений для перевірки основних параметрів системи.

Тестовий сценарій №1 - для визначення ступеня вразливості параметрів символів, що потрапляють на сайт;

Тестовий сценарій №2 - для визначення вразливості числових параметрів, що потрапляють на сайт;

Тестовий сценарій № 3 - для перевірки вразливостей поля пошуку, що дає можливість пошуку даних в базі даних;

Тестовий сценарій № 4 - для визначення вразливостей у полях авторизації;

Тестовий сценарій № 5 - для тестування та визначення вразливостей у формах реєстрації.

Тестовий сценарій № 1

Основне призначення сценарію визначити ступінь вразливості параметрів символів, що потрапляють на сайт.

Вхідні дані: будь-який параметр-символ.

Набір символів (додаткові символи):

1. «» - пусте значення
2. « ’ » - одинарна лапка
3. « ‘ ‘ » - дві одинарні лапки
4. « \ ‘’ » - дві одинарні лапки, перша з яких екранована символом backslash)
5. « -- » або « # » або « / * », « */ » (знак коментаря у SQL)
6. « “ » - подвійні лапки
7. « ‘ or 1`=`1 » - комбінація одинарної лапки зі правильним вираженням
8. « “ or “1”=”1 » - комбінація лапок з правильним вираженням
9. « ‘) or (‘1’= ‘1 » - комбінація одинарних лапок з вираженням
10. ***Складання тестового параметра за формулою:***

<Тестовий параметр>: = «; SELECT 0 {, 0} (зростаюча з кожною ітерацією кількість параметрів) > FROM {системні таблиці різних баз даних}».

11. ***Складання тестового параметра за формулою:***

<Тестовий параметр>: = «UNION SELECT 0 {, 0} (зростаюча з кожною ітерацією кількість параметрів) > FROM {системні таблиці різних баз даних} <знак коментаря в SQL>».

12. Вставка «/ ** /» на порожніх місцях у попередніх формулах.
13. Вставка «+» або ASCII-коду цього символу на місце прогалин в попередніх випадках.
14. Розглянуті вище набори символів з довільної комбінації літер і ASCII-кодів цих символів.

Тестовий сценарій № 2

Основне призначення сценарію це визначення вразливостей числових параметрів, що потрапляють на сайт.

Вхідні дані: будь-який чисельний параметр.

Набір символів (додаткові символи):

1. «» (пусте значення).
2. «or 1 = 1» (пробіл і справжнє вираз).

3. «-» або «#» або «/ *», «* /» (знак коментаря в SQL).

4. Розглянуті вище набори символів з довільної комбінації літер і ASCII кодів цих символів.

5. Вставка «/ ** /» на місці прогалин в попередніх випадках.

6. Складання тестового параметра аналогічно № 9 в тестовому сценарії 1.

7. Складання тестового параметра аналогічно № 10 в тестовому сценарії 1.

8. Вставка «+» або ASCII коду цього символу на місці прогалин в попередніх випадках.

9. Складання тестового параметра за формулою
<Тестовий параметр>: = <правильний параметр + 1> -1.

Тестовий сценарій № 3

Основне призначення сценарію - це перевірка вразливостей поля пошуку, що дає можливість пошуку даних в базі даних.

Вхідні дані: параметр, що передається в поле пошуку

Набір символів, що використовується для створення тестів:

1. «» (Пусте значення).

2. « '#» (Одинарна лапка і знак коментаря в SQL).

3. «%» (Знак довільного виразу при пошуку збігів оператором LIKE в SQL).

4. «% '#».

5. « '/ *» І «* /» для параметра sort (сортування), якщо є можливість встановлювати значення параметру sort.

Тестовий сценарій № 4

Основне призначення сценарію визначення вразливостей у полях авторизації.

Вхідні дані: параметри, що передаються у полях авторизації.

Спосіб тестування: додавання к імені користувача (login) та паролю (password) неочікуваних даних.

Набір символів, що використовується для створення тестів:

1. «<Ім'я існуючого користувача> 'or' 1 '=' 1» (ім'я існуючого користувача з істинним виразом).

2. «<ім'я існуючого користувача> '#» (ім'я існуючого користувача і знак коментаря в SQL).

3. Підстановка вищенаведених значень в поле пароля та комбінація цих значень в поле логіна і пароля.

4. «%» в поле пароля.

Тестовий сценарій № 5

Основне призначення сценарію це визначення вразливостей у формах реєстрації

Вхідні дані: параметри, що передаються в поля реєстрації.

Спосіб тестування: додавання до реєстраційних даних неочікуваних символів.

Алгоритм тестування: спроба авторизуватися користувачем, зареєстрованим з вірними параметрами, а потім користувачем з тестовими параметрами.

Набір символів, (додаткові символи):

1. складання тестового параметра за формулою:

<Тестовий параметр>: = {, <значення, підібрані евристичним чином>}) <знак коментаря в SQL>.

Примітка. У цій формулі закладено припущення, що проходження зберігаються параметрів в SQL-запиті, таке ж, як і порядок полів на формі. Значення в {} дужках повторюються стільки разів, скільки полів, починаючи від поля, що приймає поточний параметр, до кінця списку полів. Підбір евристичним чином означає, що тип параметрів оцінюється евристичним чином.

На цьому етапі тестування вам необхідно перевірити кожен зі створених тестових сценаріїв, дотримуючись схеми тестування на досліджуваному сайті для отримання повних та точних результатів тестування.

2.4.7 Оцінка результатів та формування звіту

Заключним етапом тестування буде оцінка результатів та формування звіту з результатами тестування.

Оцінка результатів чи не найважливіший етап у тестуванні. Тепер коли ви отримали результати тестування для кожного з тестового сценарію їх необхідно проаналізувати та запропонувати варіанти рішень для системи/досліджуваного сайту для унеможливлення реалізації SQL-ін'єкції.

Як зазначалося вище, для визначення та правильного аналізу результатів необхідно звертати увагу на параметр тестування, іншими словами, сценарій та тест вважається позитивним, коли параметр, що був відправлений на сайт та параметр, що надійшов з сайту на proxy має у собі додаткові символи. В такому випадку можна вважати, що параметр що тестується є вразливий до атаки за допомогою SQL-ін'єкції. Саме на такі випадки необхідно звертати увагу при розробці рекомендацій та поліпшень для системи.

Прийміть до уваги, що тестування необхідно повторювати після того, як ви обновили версію Magento, після оновлення системи та сайту новим функціоналом, після оновлення патчів безпеки та після встановлення додаткових розширень та налаштувань. Кожне з оновлення може дати кіберзлочинам додатковий шанс зламати систему.

Після додавання та реалізації нового функціоналу необхідно оновлювати та редагувати тестові сценарії, для того аби вони покривали увесь необхідний функціонал на своєчасно виявляти критичні сторінки.

Також не буде зайвим час від часу перевіряти запити, що надходять до системи. Так ви зможете контролювати підозрілу активність чи виявити спроби SQL-ін'єкцій, ще до того як вони стануть успішними.

У методиці був розроблений шаблон звіту, який можна використовувати для презентації та аналізу результатів тестування.

Шаблон звіту представлений на рисунку 2.19.

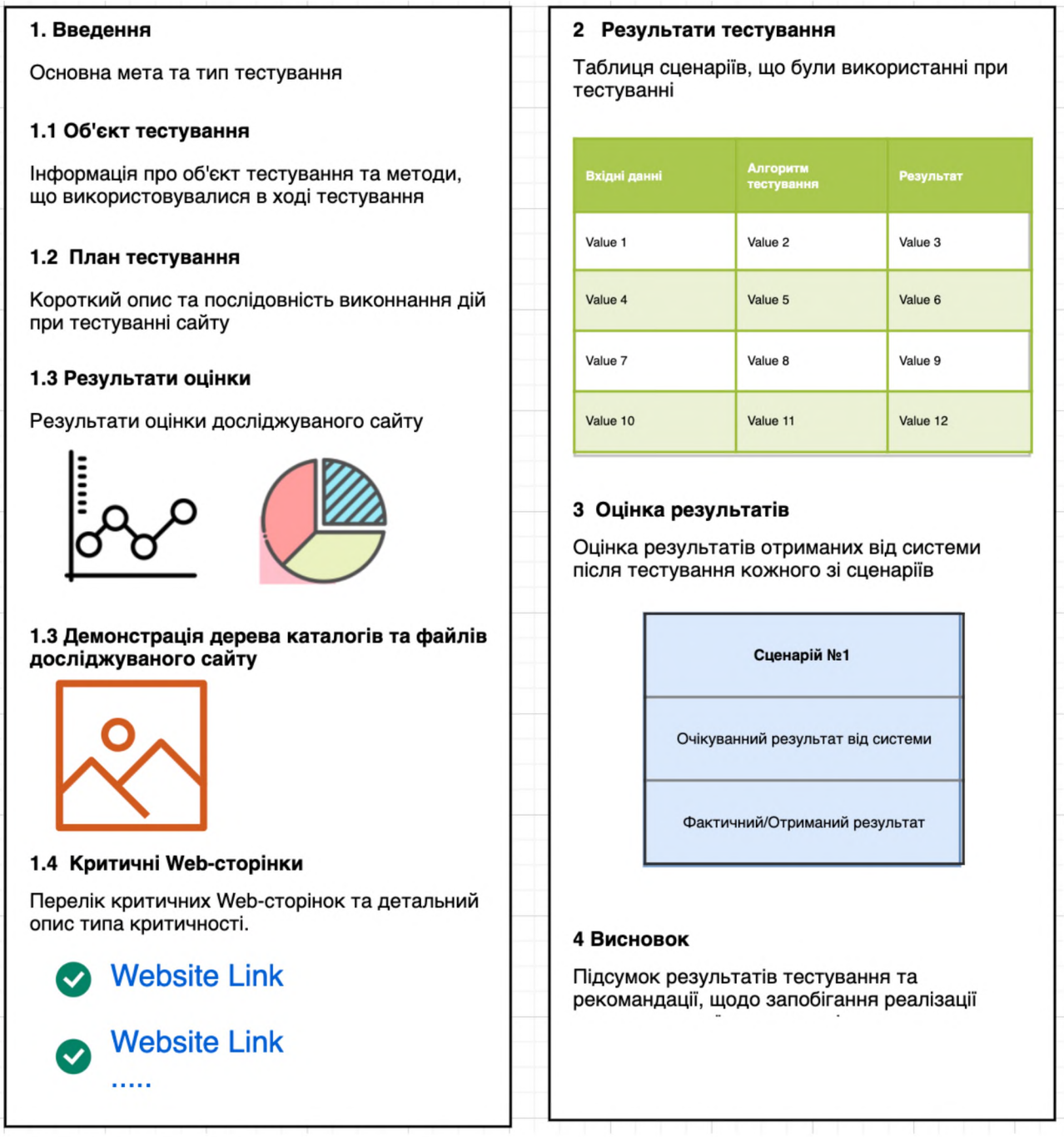


Рисунок 2.19 - Шаблон формування звіту

2.4.8 План тестування

Запропонована методика тестування та виявлення SQL-ін'єкцій для веб-сайтів розроблених за допомогою CMS Magento, дозволить протестувати та оцінити вразливість сайту до атак за допомогою SQL ін'єкцій.

Підсумовуючи вище зазначені пункти та необхідні етапи тестування, можна сформувати план тестування, що складається з таких кроків:

1. Оцінка досліджуваного сайту;

2. Створення базової середи тестування;

3. Пошук критичних WEB-сторінок;

4. Визначення критичних значень для параметрів та їх типів;

5. Підготовка та запуск тестування для знайдених сторінок та параметрів;

6. Оцінка результатів тестів та знайдених під час тестування вразливостей;

7. Формування звіту.

Використовуючи описану методику та запропонований підхід до тестування та виявлення SQL ін'єкції, можна в рази зменшити вірогідність реалізації атаки на сайт та своєчасно усувати критичні вразливості сайту.

На рисунку 2.20 зображено пошаговий план тестування та виявлення SQL-ін'єкцій для веб-сайту розробленому за допомогою CMS Magento.

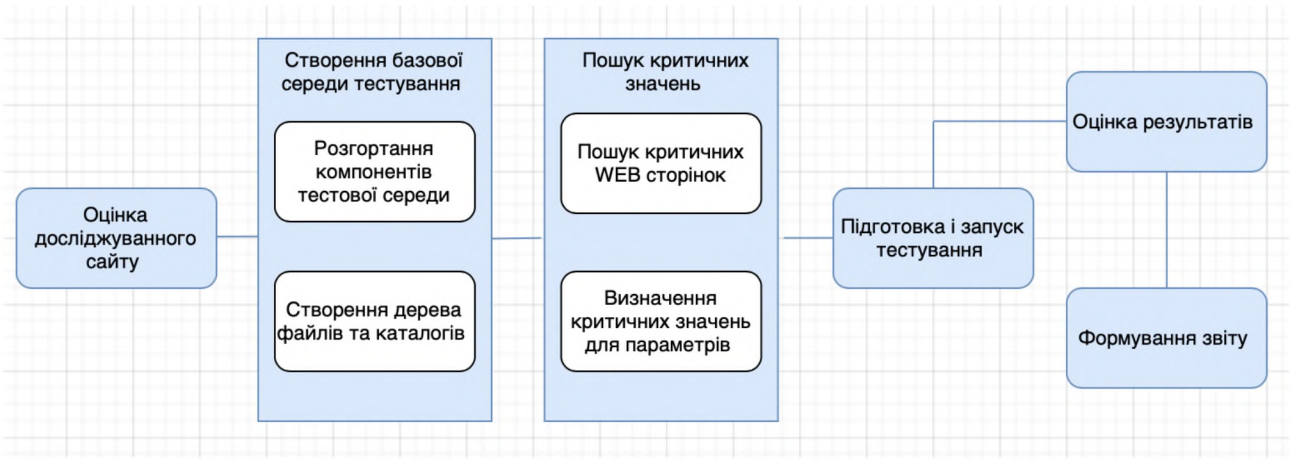


Рисунок 2.20 - План тестування сайта для виявлення SQL ін'єкцій

2.5 Висновок

У спеціальній частині кваліфікаційної роботи магістра було:

- виділено та розглянуто 10 основних вразливості сайту, які найчастіше зустрічаються при аналізі сайту та стають причинами злому хакерів згідно зі стандартом OWASP;
- наведені основні дані про SQL ін'єкції;
- описано принцип роботи SQL ін'єкції;
- розроблена методика тестування та виявлення SQL ін'єкцій для веб-сайтів розроблених на CMS Magento;

Під час розробки методики тестування та виявлення SQL ін'єкцій для веб-сайтів розроблених на CMS Magento була:

- проведена оцінка досліджуваного сайту;
- створена базова середа для тестування;
- проведено пошук на визначення критичних Web-сторінок де можлива реалізація SQL ін'єкцій;
- визначено критичні значення параметрів та їх типи;
- розроблено 5 тестових сценаріїв для запуску тестів;
- розроблено шаблон звіту для оцінки результатів проведеного тестування;
- розроблено шаблон формування тестових сценарії та схема тестування.

Використовуючи описану методику та запропонований підхід до тестування та виявлення SQL ін'єкції, можна в рази зменшити вірогідність реалізації атаки на сайт та своєчасно усувати критичні вразливості сайту.

РОЗДІЛ 3. ЕКОНОМІЧНИЙ РОЗДІЛ

3.1 Постановка задачі

Метою даного розділу є обґрунтування економічної доцільності застосування розробленої методики для веб-сайтів розроблених на CMS Magento.

В економічній частині кваліфікаційної роботи магістра буде проведено економічне обґрунтування ефективності використання розробленої методики тестування та виявлення SQL-ін'єкцій для веб-сайтів розроблених на CMS Magento, будуть проведені розрахунки капітальних та експлуатаційних витрат на реалізацію тестування та виявлення вразливості. Також будуть розраховані експлуатаційні витрати на розробку методики. Будуть проаналізовані і оцінені можливі збитки від реалізації атаки SQL-ін'єкції для інтернет-магазину розробленому за допомогою CMS Magento.

В економічній частині також буде прорахований загальний ефект від впровадження запропонованної методики. Буде визначено коефіцієнт повернення інвестицій та термін окупності капітальних інвестицій для розуміння доцільності впровадження запропонованої методики для інтернет-магазинів розроблених за допомогою CMS Magento.

3.2 Визначення капітальних та експлуатаційних витрат для розробленої методики тестування та виявлення вразливості

3.2.1 Визначення капітальних інвестицій на планування та впровадження розробленої методики тестування

До капітальних інвестицій слід віднести кошти, що необхідні на придбання та створення усіх необхідних умов для можливості запуску тестування.

Капітальні (фіксовані) витрати на придбання та створення необхідних умов для старту та запуску тестів, розраховуються за формулою:

$$К = К_{пр} + К_{зпз} + К_{пз} + К_{аз} + К_{навч} + К_{н}, \quad (3.1)$$

де $К_{пр}$ – вартість розробки персонального плану тестування та залучення для цього зовнішніх консультантів, тис. грн;

$К_{зпз}$ – вартість закупівель ліцензійного основного та додаткового програмного забезпечення (ПЗ), тис. грн;

$К_{рп}$ – вартість розробки тестових сценаріїв, тис. грн;

$К_{аз}$ – вартість закупівлі апаратного забезпечення та допоміжних матеріалів, тис. грн;

$К_{навч}$ – витрати на навчання технічних фахівців і обслуговуючого персоналу, тис. грн;

$К_{н}$ – витрати на встановлення обладнання та налагодження системи тестування, тис. грн.

До капітальних інвестицій буде включено:

- вартість розробки персонального плану тестування, в нашому випадку розробка плану тестування для досліджуваного інтернет-магазина використовуючи розроблену методику;
- витрати на залучення зовнішніх консультантів;
- вартість первісних закупівель ліцензійного основного та додаткового програмного забезпечення;
- витрати на навчання технічних фахівців і обслуговуючого персоналу.

Порахуємо витрати на первісну закупівлю ліцензійного основного та додаткового програмного забезпечення, витрати на навчання технічних фахівців і обслуговуючого персонал та витрати на залучення первісних консультантів.

При розрахунках врахуємо, що робоче місце тестувальника, вже обладнаному усіма необхідними матеріалами, додатково необхідно придбати та налаштувати Proxy сервер (3750 грн) для створення необхідної середи тестування. Консультація зовнішнього консультанта з питань інформаційної безпеки інтернет-магазинів, що складає 13 000 грн, та витрати на навчання тестувальника, що дорівнюють 9200 грн.

Отже,

$$К = 3750 + 13000 + 9200 = 25950 \text{ грн.}$$

3.2.2 Визначення витрат на розробку персонального плану тестування та тестових сценаріїв

Для визначення витрат на розробку персонального плану тестування та тестових сценаріїв буде розрахована трудомісткість розробки плану для одного інтернет-магазину та проведено розрахунок витрат.

3.2.2.1 Визначення трудомісткості розробки персонального плану тестування та тестових сценаріїв

Трудомісткість розробки персонального плану тестування та тестових сценаріїв визначається тривалістю кожної робочої операції, починаючи з складання технічного завдання і закінчуючи оформленням документації та звітності (за умови роботи одного спеціаліста з інформаційної безпеки в нашому випадку тестувальника безпеки) та обчислюється за формулою:

$$t = t_{тз} + t_{в} + t_{а} + t_{вз} + t_{озб} + t_{овр} + t_{д}, \text{ годин,} \quad (3.2)$$

де $t_{тз}$ – тривалість складання персонального плану тестування для інтернет-магазину, з урахуванням його особливостей;

$t_{в}$ – тривалість розробки 1 тестового сценарію;

$t_{а}$ – тривалість процесу аналізу ризиків;

$t_{вз}$ – тривалість визначення вимог до заходів, методів та засобів захисту;

$t_{озб}$ – тривалість вибору основних рішень з забезпечення безпеки інформації;

$t_{овр}$ – тривалість організації виконання відновлювальних робіт і забезпечення неперервного функціонування тестуючої системи;

$t_{д}$ – тривалість документального оформлення плану тестування.

Отже, $t = 12 + 5x2 + 2 + 1 + 16 + 10 = 51$ година.

3.2.2.2. Розрахунок витрат на тестування запропонованої методики для одного інтернет-магазину

Витрати на тестування запропонованої методики для 1 інтернет-магазину ***Крп*** складаються з витрат на заробітну плату спеціаліста з інформаційної безпеки (в нашому випадку тестувальника безпеки) ***Ззп*** і вартості витрат машинного часу, що необхідний для тестування ***Змч***:

$$К_{рп} = З_{зп} + З_{мч}\ , \qquad (3.3)$$

Заробітна плата виконавця враховує основну і додаткову заробітну плату, а також відрахування на соціальне потреби (пенсійне страхування, страхування на випадок безробіття, соціальне страхування тощо) и визначається за формулою:

$$Ззп = t * Зіб, грн, \qquad (3.4)$$

де t – загальна тривалість тестування, годин;

Зіб – середньогодинна заробітна плата спеціаліста з інформаційної безпеки з нарахуваннями, грн/годину.

$$Ззп = 32 * 400 = 12\ 800 грн.$$

Вартість машинного часу для проведення тестування на робочій станції співробітника визначається за формулою:

$$Змч = t * Смч, грн, \qquad (3.5)$$

де t – трудомісткість проведення тестування на робочій станції співробітника, годин;

$C_{мч}$ – вартість 1 години машинного часу ПК, грн./година.

Вартість 1 години машинного часу ПК визначається за формулою:

$$C_{мч} = P \times t_{нал} \times C_e + \frac{Ф_{зал} \times Ha}{Fp} + \frac{K_{лпз} \times H_{апз}}{Fp}, грн, \qquad (3.6)$$

де Р – встановлена потужність ПК, кВт;

$С_е$ − тариф на електричну енергію, грн/кВт·година;

$Ф_{зал}$ – залишкова вартість ПК на поточний рік, грн.;

$Н_а$ – річна норма амортизації на ПК, частки одиниці;

$Н_{апз}$ – річна норма амортизації на ліцензійне програмне забезпечення, частки одиниці;

$К_{лпз}$ – вартість ліцензійного програмного забезпечення, грн.;

$F_р$ – річний фонд робочого часу (за 40-годинного робочого тижня, $F_р = 1920$).

Залишкова вартість ПК визначається виходячи з фактичного терміну його експлуатації як різниця між первісною вартістю та зносом за час використання.

Використаємо прямолінійний метод амортизації, за яким річна сума амортизації визначається діленням вартості, яка амортизується, на строк корисного використання об'єкта основних засобів.

$$С_{мч} = 0,238 * 1,68 + 4,6 + 0,99 = 5,9 \text{ грн.}$$

Тому можна порахувати вартість машинного часу для проведення тестування на ПК, що буде становити:

$$З_{мч} = 51 * 5,9 = 300,9 \text{ грн.}$$

Таким чином, капітальні (фіксовані) витрати на проведення тестування 1 інтернет-магазину розробленому на CMS Magento складають:

$$К = 25950 + 12\,800 + 300,9 = 39050,9 \text{ грн.}$$

3.3. Розрахунок експлуатаційних витрат для підтримки тестування та проведення повторного тестування, в разі необхідності

Для визначення витрат на підтримку тестування та проведення його повторно, в разі необхідності скористаємося формулою для розрахунку:

$$С = Сн + Сз + Сел + Стос, грн, \quad (3.7)$$

де Сн - витрати на навчання працівників, що будуть проводити тестування інтернет-магазину

Сз - річний фонд заробітної плати працівників та інженерно-технічного персоналу, що обслуговує систему

Сел – вартість електроенергії, що споживається апаратурою для забезпечення безперебійної роботи інтернет-магазину протягом року

Стос – витрати на технічне й організаційне адміністрування та сервіс системи.

Витрати на навчання працівників відділу та системного адміністратора для проведення додаткового тестування складають 15 600 грн.

Для того, щоб розрахувати річний фонд заробітної плати працівників та інженерно-технічного персоналу, що обслуговує систему, скористаємося формулою для розрахунку:

$$Сз = 12 * (Зосн + Здод), грн, \quad (3.8)$$

де Зосн - основна заробітна плата;

Здод - додаткова заробітна плата.

(Додаткова заробітна плата визначається як 10% від основної заробітної)

До річного фонду заробітної плати додається єдиний внесок на загальнообов'язкове державне соціальне страхування у розмірі 22% від суми основної та додаткової заробітної плати.

Тому маємо:

$$Сз = 12 * (23000 + 23000 * 0,01 + 0,22 * 23000) = 339480 грн.$$

Розрахуємо вартість електроенергії, що споживається апаратурою для забезпечення безперебійної роботи інтернет-магазину протягом року за формулою:

$$Сел = Р \cdot Fр \cdot Це, грн, \quad (3.9)$$

де Р – встановлена потужність апаратури інформаційної безпеки, кВт;

Fр – річний фонд робочого часу системи інформаційної безпеки;

Це – тариф на електроенергію, грн/кВт · годин.

$$Сел = 5 * 0,238 * 1920 * 1,68 = 3838 \text{ грн.}$$

Витрати на технічне й організаційне адміністрування та сервіс системи в середньому витрачаються як 2% від суми капітальних витрат.

$$Стос = 0.02 * 39050,9 = 781 \text{ грн.}$$

Порахувавши витрати окремо, можемо визначити загальну суму витрат на підтримку тестування:

$$С = 15600 + 339480 + 3838 + 781 = 359\ 699 \text{ грн.}$$

3.4 Оцінка можливого збитку від реалізації атаки SQL ін'єкції на інтернет-магазин

Для того аби оцінити можливий збиток від реалізації атаки SQL ін'єкції на інтернет-магазин потрібно розрахувати величину упущеної вигоди від простою атакованого вузла або сегмента корпоративної мережі. Що рахується за формулою:

$$U = Пп + Пв + V, \text{ грн,} \qquad (3.10)$$

де Пп – оплачувані втрати робочого часу та простою співробітників атакованого вузла або сегмента корпоративної мережі, грн;

Пв – вартість відновлення працездатності вузла або сегмента корпоративної мережі (переустановлення системи, зміна конфігурації та ін.), грн;

V – втрати від зниження обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі, грн.

Розрахуємо витрати від зниження продуктивності співробітників атакованого вузла або сегмента корпоративної мережі:

$$П_п = \frac{ЗЗс \times Чс}{F} \times t_п, \text{ грн}, \quad (3.11)$$

де Зс – місячна заробітна плата співробітника атакованого вузла або сегмента корпоративної мережі з нарахуванням єдиного соціального внеску, грн на місяць;

Чс – чисельність співробітників атакованого вузла або сегмента корпоративної мережі, осіб.;

tп – час простою вузла або сегмента корпоративної мережі внаслідок атаки, год

F – місячний фонд робочого часу.

Тому маємо:

$$П_п = \frac{17\,000 \times 3 + 25\,000 + 40\,000}{16} \times 16 = 11600 \, грн.$$

Витрати на відновлення працездатності вузла або сегмента корпоративної мережі включають кілька складових розраховуються наступним чином:

$$Пв = Пви + Ппв, \text{ грн}, \quad (3.12)$$

де Пви – витрати на повторне уведення інформації, грн;

Ппв – витрати на відновлення вузла або сегмента корпоративної мережі, грн.

Розрахуємо витрати на повторне уведення інформації за формулою:

$$Пви = \frac{ЗЗс \times Чс}{F} \times t_{ви}, грн, \quad (3.13)$$

де tви – час повторного введення загубленої інформації співробітниками атакованого вузла або сегмента корпоративної мережі, годин.

Пви = 725 * 25 = 18124 грн.

Та витрати на відновлення вузла або сегмента корпоративної мережі:

$$П_{пв} = \frac{\Sigma Зо \times Чс}{F} \times t_в, грн, \quad (3.14)$$

де tв – час відновлення вузла після атаки персоналом, що обслуговує корпоративну мережу, годин;

Ппв = 725 * 15 = 10875 грн.

Втрати від зниження очікуваного обсягу продажів за час простою атакованого вузла або сегмента корпоративної мережі будуть пораховані за формулою:

$$V = \frac{O}{Fг} \times (t_п + t_в + t_и), грн, \quad (3.15)$$

де О – обсяг чистого прибутку від фінансової діяльності інтернет магазину;

Fг – річний фонд часу роботи організації.

Річний фонд робочого часу при 8-ми годинному робочому дні складає 1920 годин.

Fг = 1920 годин.

Отже,

$$V = \frac{86163}{1920} \times (15 + 25 + 16) = 2508,8 \text{ грн.}$$

Маємо, що величина упущеної вигоди від простою атакованого вузла або сегмента корпоративної мережі буде дорівнювати:

U = 18124 + 10875 + 2508, 8 = 31507,8 грн.

Порахуємо загальний збиток від атаки на вузол або сегмент корпоративної мережі організації:

$$B = \Sigma U \times N \times I, грн, \quad (3.16)$$

де N – середнє число можливих атак на рік;

I – число атакованих вузлів або сегментів корпоративної мережі.

B = 31507,8 * 2 * 5 = 315078 грн.

Загальний ефект від впровадження запропонованої методики тестування з урахуванням ризиків порушення інформаційної безпеки інтернет-магазину становить:

$$E = B * R - C, грн, \quad (3.17)$$

де B – загальний збиток від атаки на вузол або сегмент корпоративної мережі, тис. грн;

R – очікувана імовірність атаки на вузол або сегмент корпоративної мережі, частки одиниці;

C – щорічні витрати на експлуатацію та підтримку методики, тис. грн.

E = 315078 * 2 - 359 699 = 270 457 грн.

3.5 Визначення показників економічної ефективності методики тестування

Коефіцієнт повернення інвестицій ROSI показує, скільки гривень додаткового прибутку приносить одна гривня капітальних інвестицій на впровадження методики тестування.

$$ROSI = \frac{E}{K}, \text{частки одиниці}, \quad (3.18)$$

де E – загальний ефект від впровадження методики тестування та виявлення, тис. грн;

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

$$ROSI = \frac{270\,457}{39050,9} = 6,9.$$

Термін окупності капітальних інвестицій То показує, за скільки років капітальні інвестиції окупляться за рахунок загального ефекту від впровадження розробленої методики:

$$\text{То} = \frac{1}{6,9} = 0,14 \text{ року}.$$

3.6 Висновок

В економічному розділі було визначено капітальні (фіксовані) витрати на придбання та створення необхідних умов для старту та запуску тестів, що складають – 39050,9 грн.

Розрахунок експлуатаційних витрат для підтримки тестування та проведення його повторно, в разі необхідності, що складають - 359 699 грн грн у рік.

Проведена оцінка можливого збитку від реалізації атаки SQL ін'єкції на інтернет-магазин розроблений за допомогою CMS Magento. При можливості атак двічі на рік загальний збиток від атаки буде становити 315 078 грн у рік.

Коефіцієнт повернення інвестицій ROSI дорівнює 6,9.

А загальний ефект від впровадження системи інформаційної безпеки буде складати 270 457 грн, що свідчить про те, що запроповані засоби дозволять значно знизити рівень збитку від атак та не допустити його.

Як видно з розрахунків використовувати запропоновану методику доцільно, а капітальні витрати повернуться через 51 день.

ВИСНОВКИ

У кваліфікаційній роботі магістра було розглянуто поняття CMS системи, її основне призначення та особливості, бо необхідність чіткого розуміння та надійного вибору CMS системи для бізнесу постає на перший план. Потрібно ще з моменту ідеї розробки сайту підійти дуже ретельно до вибору CMS системи та необхідного рівня безпеки для його реалізації, тому основна увага у першому розділі роботі була приділена саме на технічні особливості та різницю у характеристиках основних CMS систем.

У спеціальній частині роботи були виділені і розглянуті основні вразливості сайтів розроблених на різних CMS системах які найчастіше стають причиною злому сайту. Основна увага була приділена SQL-ін'єкціям їх принципу роботи та особливостями.

У спеціальній частині роботи була розроблена та запропонована методика тестування та виявлення SQL-ін'єкцій для веб-сайтів розроблених за допомогою CMS Magento. Пошагова інструкція, шаблони тестових сценаріїв та шаблони звітів наведені у роботі. Для того аби протестувати сайт за допомогою розробленої методики та виявити вразливість був обраний тестовий інтернет-магазин розроблений за допомогою CMS Magento.

В економічній частині роботи було розраховано витрати на впровадження розробленої методики та загальні збитки інтернет-магазину від реалізації SQL атак у рік. Було доведено, що впровадження запропонованої методики тестування та виявлення є економічно доцільним, а термін окупності становить 51 день.

Тому у даному випадку є доцільним прийняти до уваги та використовувати розроблену методику тестування на виявлення для інтернет-магазинів розроблених на CMS Magento.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке CMS система [Електронний ресурс] // Kinsta. – 2020. – Режим доступу до ресурсу: https://kinsta.com/knowledgebase/content-management-system/.

2. Порівняння 15-ти найпопулярніших CMS систем у 2020 році [Електронний ресурс] // WPbeginners. – 2020. – Режим доступу до ресурсу: https://www.wpbeginner.com/showcase/best-cms-platforms-compared/.

3. Content Management Systems (CMS) [Електронний ресурс] // TrustRadios. – 2020. – Режим доступу до ресурсу: https://www.trustradius.com/cms.

4. Принцип роботи та архітектура CMS систем [Електронний ресурс] // Sam Solution. – 2020. – Режим доступу до ресурсу: https://www.sam-solutions.com/blog/what-is-a-cms/.

5. Принципи розробки самописних CMS систем [Електронний ресурс] // Fandom. – 2020. – Режим доступу до ресурсу: https://cms.fandom.com/wiki/Principles_of_creating_a_CMS.

6. Принципи розробки систем керування вмістом сайтів [Електронний ресурс] // Портал знань. – 2020. – Режим доступу до ресурсу: http://www.znannya.org/?view=cms-development.

7. Статистика злому сайтів розроблених за допомогою різних CMS систем [Електронний ресурс] // Astra. – 2019. – Режим доступу до ресурсу: https://www.getastra.com/blog/cms/hacking-statistics/.

8. НД ТЗІ 1.1-002-99: Загальні положення з захисту інформації в комп'ютерних системах від НСД (введено в дію Наказом ДСТСЗІ СБУ від 28.04.1999 р. No 22) [Електронний ресурс] – Режим доступу до ресурсу: https://tzi.com.ua/downloads/1.1-002-99.pdf

9. НД ТЗІ 1.1-003-99: Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу [Електронний ресурс] – Режим доступу до ресурсу: http://www.dut.edu.ua/uploads/l_1021_47029323.pdf.

10. НД ТЗІ 2.5-004-99: Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу (введено в дію Наказом

ДСТСЗІ СБУ від 28.04.1999 р. No 22). [Електронний ресурс] – Режим доступу до ресурсу: https://tzi.ua/assets/files/НД-ТЗІ-2.5-004-99.pdf.

11. НД ТЗІ 2.5-010-03 Вимоги до захисту інформації WEB-сторінки від несанкціонованого доступу (прийнято 2003-04-02) [Електронний ресурс] – Режим доступу до ресурсу: https://inspections.gov.ua/document/view?id=374.

12. Стандарт захисту даних платіжних карток (PCI DSS) [Електронний ресурс] – Режим доступу до ресурсу: http://iso27000.ru/standarty/pci-dss-standart-zaschity-informacii-v-industrii-platezhnyh-kart-1.

13. ДСТУ ISO / IEC TR 20004: 2017 Інформаційні технології. Методи захисту. Уточнений аналіз вразливості програмного забезпечення відповідно до ISO / IEC 15408 та ISO / IEC 18045 (ISO / IEC TR 20004: 2015 року, IDT) [Електронний ресурс] – Режим доступу до ресурсу: http://online.budstandart.com/ru/catalog/doc-page.html?id_doc=74704.

14. 10 уязвимостей сайтов и методы их устранения [Електронний ресурс] // Генератор продаж. – 2020. – Режим доступу до ресурсу: https://sales-generator.ru/blog/uyazvimosti-saytov/

15. OWASP Top Ten [Електронний ресурс] // OWASP. – 2020. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/

16. WhatWeb & Wappalyzer Scan [Електронний ресурс] // Hacker Target. - 2020 – Режим доступу до ресурсу: https://hackertarget.com/whatweb-scan/

17. SUCURI [Електронний ресурс] //SUCURI SITECHECK. - 2020 – Режим доступу до ресурсу: https://sitecheck.sucuri.net/results/https/m233.dev-dinarys.com

18. OWASP [Електронний ресурс] //WSTG-Latest. - 2020 – Режим доступу до ресурсу: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

19. Magento SQL injection: How to secure your Magento store against SQL-injection attack [Електронний ресурс] //ASTRA. - 2020 – Режим доступу до ресурсу: https://www.getastra.com/blog/cms/magento-security/magento-sql-injection/

20. How to test SQL injection types manually [Електронний ресурс] // Habilelabs . - 2020 – Режим доступу до ресурсу: https://www.habilelabs.io/sql-injection-types-how-to-test-sql-injection-manually/

21. Тестирование сайтов и приложений [Електронний ресурс] // GetBug . - 2018 – Режим доступу до ресурсу: http://getbug.ru/testirovanie-bezopasnosti-saytov-i-prilozheniy/

22. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів спеціальності 125 Кібербезпека/ Упоряд.: О.В. Герасіна, Д.С. Тимофєєв, О.В. Кручінін, Ю.А. Мілінчук – Дніпро: НТУ «ДП», 2020. – 47 с.

ДОДАТОК А. Відомість матеріалів кваліфікаційної роботи

№	Формат	Найменування	Кількість листів
1	А4	Реферат	3
2	А4	Список умовних скорочень	1
3	А4	Зміст	2
4	А4	Вступ	2
5	А4	Розділ 1. Стан питання. Постановка задач	17
6	А4	Розділ 2. Спеціальна частина	37
7	А4	Розділ 3. Економічна частина	11
8	А4	Висновки	1
9	А4	Перелік посилань	3
10	А4	Додаток А	1
11	А4	Додаток Б	1
12	А4	Додаток В	1
13	А4	Додаток Г	1
14	А4	Додаток Ґ	1
15	А4	Додаток Д	12
16	А4	Додаток Е	1
17	А4	Додаток Є	1
18	А4	Додаток Ж	1

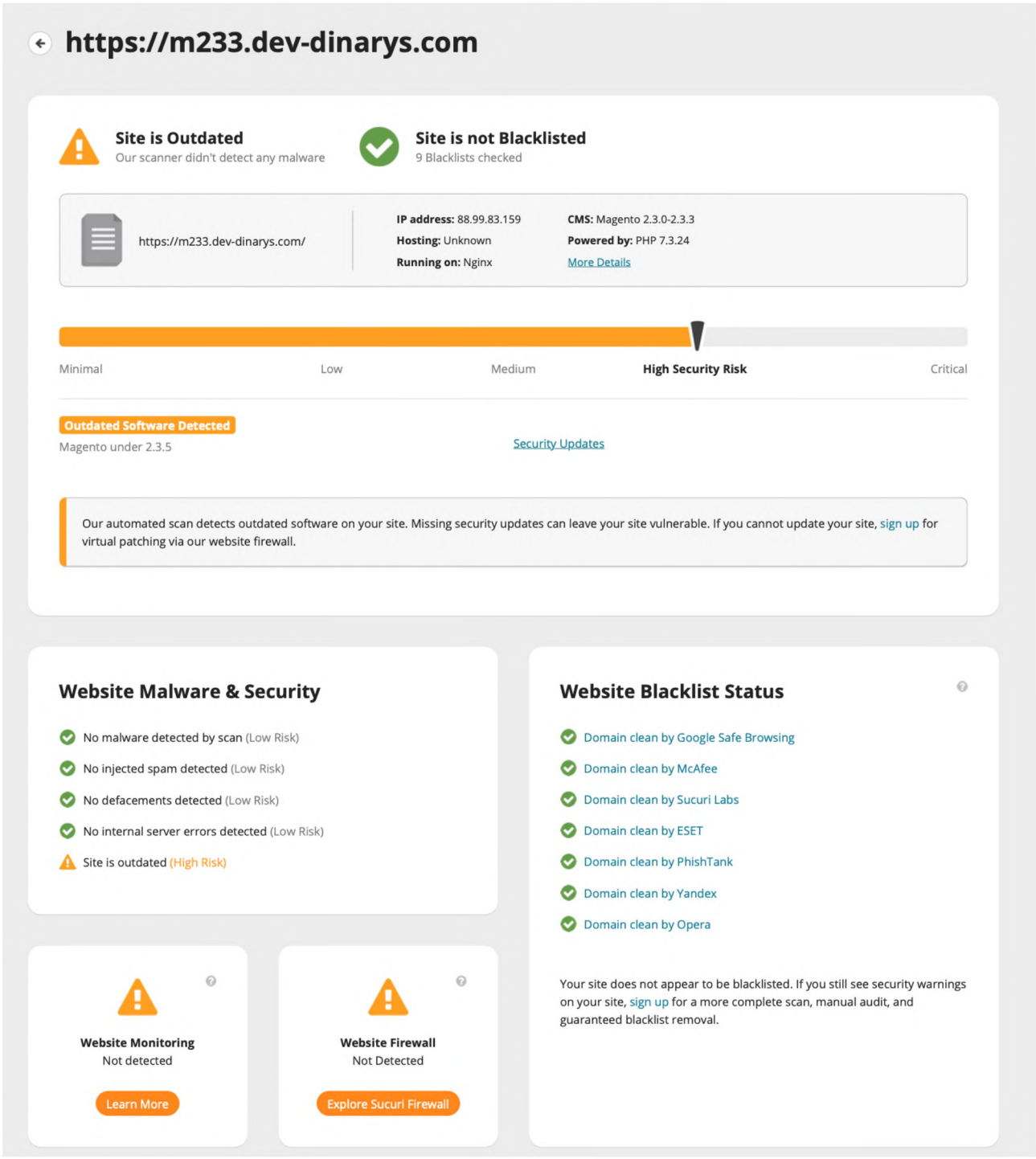
ДОДАТОК Б. Порівняльна характеристика основних CMS систем частина 1

Порівняльна характеристика основних CMS систем		
	Переваги	Недоліки
WordPress.org	1. Пропонує гнучкість і свободу створення будь-якого веб-сайту 2. Не потребує знання у кодуванні чи технічних знань 3. Повна свобода заробляти гроші онлайн будь-яким способом 4. Доступні тисячі тем і плагінів, як безкоштовних так і платних. 5. Має дуже гнучкі можливості SEO оптимізації 6. Має можливість завантажити весь ваш контент у XML форматі	1. Необхідно налаштувати хостинг та домен 2. Вам потрібно самим бути відповідальним за налаштування безпеки та бєкапів 3. Через велику кількість функцій та можливостей, для початківців може бути скадно розібратися
Joomla	1. Пропонує гнучкість і безлічь налаштуваннь. Це хороший вибор, якщо потрібно розробити щось складне, а бо кастомне 2. Хороший інструмент як для розробників, так і для тих хто не вміє програмувати 3.Як і WordPress, Joomla є відкритим кодом, і якщо ви застрягнете, доступна велика спільнота підпримки 4. Може використуватися як e-commerse магазин, якщо встановити додаткові плагіни	1. Навіть шанувальники Joomla визнають, що вона може бути досить скадною і може знадобитись допомога розробника. 2. Існує так багато розширень та додаткових плагінів 3.Якщо встановлено багато додаткових модулів та плагінів, можуть виникнути проблеми зі сумістністю
Drupal	1. Легке додавання контенту 2. Велика кількість додаткових модулів 3.Доступна велика спільнота підпримки 4. Можливість створення різні ролі користувачів та їх рівень доступу	1. Може бути складно зрозуміти як змінити зовнішній вигляд сайту 2. Більшість веб-сайтів мають спеціально налаштовану тему, створенну розробником

ДОДАТОК В. Порівняльна характеристика основних CMS систем частина 2

Woocommerce	1. Це безкоштовне програмне забезпечення, щоб почати потрібен хостинг та домен 2. Велика кількість готових тем для веб-сайту, що дозволяє розробити сайт саме таким, як потрібно 3. Дозволяє продавати різні види товарів на сайт 4. Має інструменти для управління інвентаризацією 5. Має вбудовану можливість оплати використовуючи PayPal та Stripe.	1. WooCommerce технічно працює з будь-якою темою WordPress, але ви можете дотримуватися тем, створених спеціально для WooCommerce для розширеної підтримки. 2. WooCommerce у налаштуванні може бути складним для новачків
Shopify	1. Ви можете приймати кредитні та дебетові картки за допомогою інтегрованого платіжного рішення Shopify, Shopify Payments. 2. Доступно безліч розширень та тем. Ви можете придбати сторонні програми Shopify, які дозволяють додавати всілякі функції до вашого Інтернет-магазину. 3.Має цілодобову підтримку через чат, електронну пошту, телефон і Twitter. Доступна також велика кількість документації (включаючи	1. Ваші витрати можуть бути досить високими, особливо якщо ви хочете додати багато сторонніх програм у свій магазин. 2. Можливо, ви захочете додати функціональність, яка просто недоступна: програми Shopify обмежені, ніж такі речі, як плагіни WordPress.
Magento	1. Magento є дуже кастомізованою, з великою кількістю сторонніх розширень, які ви можете використовувати для додавання додаткових функцій. 2. Існує кілька справді великих брендів, що використовують Magento, серед яких Nike, Ford та Coca Cola. 3. Ви можете підключити різні платіжні шлюзи до Magento	1. Знайти хороших розробників для проектів Magento може бути складно, а найняти їх може бути дуже дорого. 2. Доступна підтримка може відрізнятися, особливо якщо ви використовуєте Magento Open Source і покладаєтесь на онлайн-форуми та допомогу.
Bitrix24	1. Базовий рівень Bitrix24 безкоштовний, тобто ви можете спробувати його, нічого не плативши 2. Бітрікс24 включає величезну кількість функцій, що дають вам все необхідне для управління невеликою та середньою компанією. 3. Хостинг вашого веб-сайту безкоштовний (якщо ви берете безкоштовний тариф).	1.Bitrix24 призначений для використання як CRM, тому, якщо у вас вже є CRM, який вас влаштовує, або ви не хочете використовувати цю функцію, це досить складний спосіб отримати платформу CMS. 2. Оскільки функцій так багато, вам може здатися, що інтерфейс Bitrix24 заплутаний або складний в навігації.

ДОДАТОК Г. Результат тестування сайту частина 1



ДОДАТОК Ґ. Результат тестування сайту частина 2

Links found

/
/customer/account/login/referer/aHR0cHM6Ly9tMjMzLmRldi1kaW5hcnlzLmNvbS8,/
/customer/account/create/
/checkout/cart/
/catalogsearch/advanced/
/what-is-new.html
/women.html
/women/tops-women.html
/women/tops-women/jackets-women.html
/women/tops-women/hoodies-and-sweatshirts-women.html
/women/tops-women/tees-women.html
/women/tops-women/tanks-women.html
/women/bottoms-women.html
/women/bottoms-women/pants-women.html
/women/bottoms-women/shorts-women.html
/men.html
/men/tops-men.html
/men/tops-men/jackets-men.html
/men/tops-men/hoodies-and-sweatshirts-men.html
/men/tops-men/tees-men.html
/men/tops-men/tanks-men.html
/men/bottoms-men.html
/men/bottoms-men/pants-men.html
/men/bottoms-men/shorts-men.html
/gear.html
/gear/bags.html
/gear/fitness-equipment.html
/gear/watches.html
/training.html
/training/training-video.html
/sale.html
/collections/yoga-new.html
/promotions/pants-all.html
/promotions/tees-all.html
/collections/erin-recommends.html
/collections/performance-fabrics.html
/collections/eco-friendly.html
/breathe-easy-tank.html
/radiant-tee.html
/argus-all-weather-tank.html
/hero-hoodie.html
/lifelong-fitness-iv.html
/push-it-messenger-bag.html
/fusion-backpack.html
/about-us/
/customer-service/
/search/term/popular/
/privacy-policy-cookie-restriction-mode/
/sales/guest/form/
/contact/

ДОДАТОК Д.

МЕТОДИКА ТЕСТУВАННЯ ТА ВИЯВЛЕННЯ SQL-ІН'ЄКЦІЙ ДЛЯ ВЕБ-САЙТІВ НА CMS MAGENTO

Методика тестування та виявлення SQL ін'єкції для веб-сайтів на CMS Magento

Призначення

Дана методика містить рекомендації, щодо проведення тестування та виявлення SQL-ін'єкцій для веб-сайтів на CMS Magento. Методика може бути використана для первісного тестування нової системи, так і для подальшої підтримки необхідного рівня безпеки.

Загальна термінологія

Загроза – це можлива причина небажаного інциденту, який може завдати шкоду системі чи організації.

Вразливість представляє собою слабке місце активу чи засобу управління, яке може бути використано однією та більше загрозою.

Атака – спроба реалізації загрози.

Примітка.

Для всіх тестових сценаріїв, запропонованих у методиці вважати:

- під справжнім вираженням - будь-яке справжнє вираження, наприклад, (1, 'a'='a', 1<>2, 3>2, 1+1, ISNULL(NULL), 2 IN (0,1,2), 2 BETWEEN 1 AND 3);
- під системними таблицями баз даних - таблиці, що містять метадані для бази даних, що тестується;
- під ім'ям існуючого користувача - ім'я користувача, що було підібрано за словарем.
- під знаком коментаря в SQL - такі знаки (# , /**/, --)

Схема тестування

Запропонована методика тестування та виявлення SQL-ін'єкцій для веб-сайтів розроблених за допомогою CMS Magento, дозволить протестувати та оцінити вразливість сайту до атак за допомогою SQL ін'єкцій.

План тестування складається з таких кроків:

1. Оцінка досліджуваного сайту;
2. Створення базової середи тестування;
3. Пошук критичних WEB-сторінок;

4. Визначення критичних значень для параметрів та їх типів;
5. Підготовка та запуск тестування для знайдених сторінок та параметрів;
6. Оцінка результатів тестів та знайдених під час тестування вразливостей;
7. Формування звіту.

Використовуючи описану методику та запропонований підхід до тестування та виявлення SQL ін'єкції, можна в рази зменшити кількість атак на сайт та своєчасно усувати критичні вразливості сайту.

На рисунку 1 зображено пошаговий план тестування та виявлення SQL-ін'єкцій для веб-сайту розробленому за домогою CMS Magento.

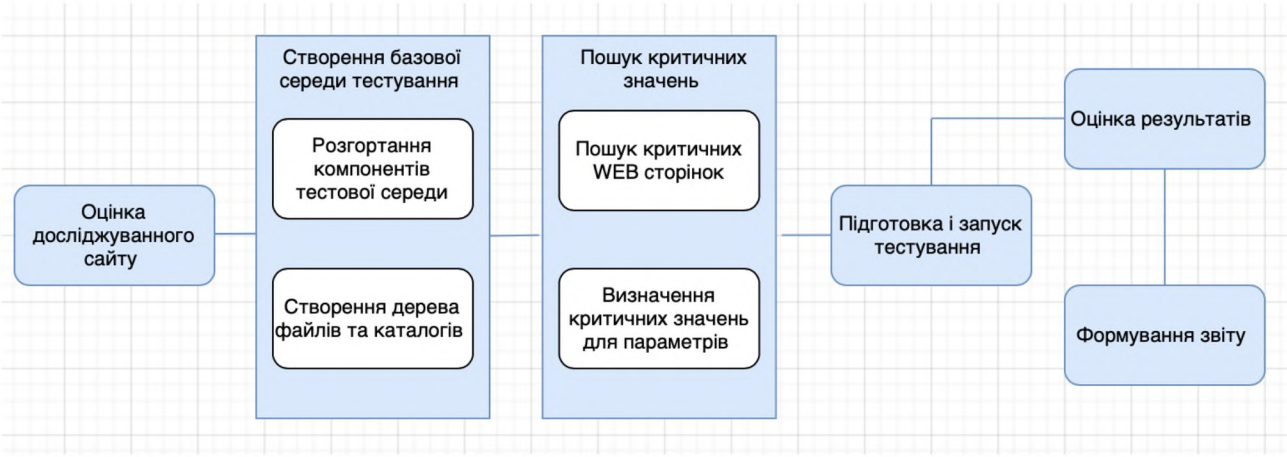


Рисунок 1 - План тестування сайта для виявлення SQL ін'єкцій

Рекомендується починати тестування з оцінки досліджуваного сайту. На цьому етапі виявимо особливості сайту, мову програмування, точну версію CMS системи, що використовується і тд [3].

Як підсумок після проведення першого етапу тестування ви повинні отримати інформацію про сайт, а саме:

- CMS система
- Версія CMS системи
- Мова програмування
- Веб-інструменти
- IP-адреса
- Хостинг

- Сервер
- Розташування домену
- Список додаткових модулів та застосунків, що використовуються в системі
- Присутній чи кастомний код на сайті
- Чи були зміни у логіці CMS Magento

Отримавши всю зазначену вище інформацію, можна переходити до другого етапу тестування - “Створення базової середи тестування”.

На цьому етапі необхідно виконати розгортку компонентів тестової середи та створити дерево файлів та каталогів. Іншими словами визначити внутрішню структуру сайту та досліджуваної системи. Необхідно визначитись зі структурою бази даних досліджуваного сайту для того, аби далі правильно та чітко сформувати тестові сценарії.

Як результат після виконання другого етапу тестування повинна бути готова на налаштована тестова середа для проведення тестування, а також досліджено і сформовано дерево файлів та каталогів досліджуваного сайту. Що на наступних етапах буде необхідно для правильного формування тестових сценаріїв та пошуку критичних Web-сторінок.

На третьому етапі тестування необхідно знайти, проаналізувати та виявити Web-сторінки, які можуть мати параметри критичні по відношенню до SQL-ін’єкції [2].

В першу чергу це сторінки, які запитують у вас дані. Це можуть бути сторінки пошуку, різних обговорень, відгуків і тд.

Інколи html сторінки використовують метод POST, щоб відправити команди з іншої Web-сторінки. В такому випадку, ви не побачите параметри в URL, але ви можете шукати тег “FORM” в коді HTML сторінки [4].

Також спробуйте знайти сторінки типу ASP, JSP, CGI або PHP Web сторінки. Вразливі до введення SQL-ін’єкцій можуть бути сторінки, що використовують параметри типу:

https://m233.dev-dinarys.com/?ID=12345

Як результат проведення третього етапу тестування у вас повинен бути сформований список потенційно критичних Web-сторінок, через які можлива реалізація SQL-ін'єкції.

Четвертий етап тестування представляє собою аналіз та визначення очікуваних параметрів та їх типів для кожної з виявленої сторінки на попередньому етапі тестування. На цьому етапі вам необхідно проаналізувати та визначити параметри, які система очікує та отримує у разі коректної роботи системи. Визначенні параметри стануть у нагоді при формування тестових сценаріїв для проведення тестування.

За допомогою визначення критичних параметрів та їх типів буде необхідно сформувати тестові сценарії, а саме комбінацію очікуваних параметрів системи з неочікуваними.

Як результат після аналізу та визначення у вас буде сформований список очікуваних параметрів для кожної зі сторінки, що була визначена як критичною для досліджуваного сайту. Визначенні параметри будуть необхідні для правильного формування тестових сценаріїв та проведення безпосереднього тестування.

Наступний, п'ятий етап тестування включає в себе підготовку до запуск самого тестування.

Тестування сайту для виявлення SQL ін'єкцій буде включати в собі тестуючий додаток (що імітує дії користувача) та створює і запускає тести та сайт, що тестується.

Далі, в схемі повинен бути налаштований proxy сервер, що імітує базу даних. В такому випадку, взаємодія сайту с базою даних буде відбуватись через цей компонент і дозволить провести більш глибоке та якісне тестування системи. Якщо ж немає змоги встановити та налаштувати proxy сервер, то зв'язок з базою даних буде прямий, а це може впливати та аналіз відповідей від системи. Бо, наявність у схемі тестування proxy сервера дозволяє підвищити точність тестування. Proxy сервер, отримує SQL запит від сайту, оцінює його та передає серверу бази даних, такий спосіб є більш точним та інформативним при проведенні тестування та виявлення вразливостей [1].

На рисунку 2 зображена схема тестування, що повинна бути створена для правильного запуску тестування.

Закінчуючи п’ятий етап тестування у вас повинна бути створена на налаштована правильна схема тестування, що складається з тестуючого додатку, серверу, Proxy серверу та бази даних згідно рисунку 2.

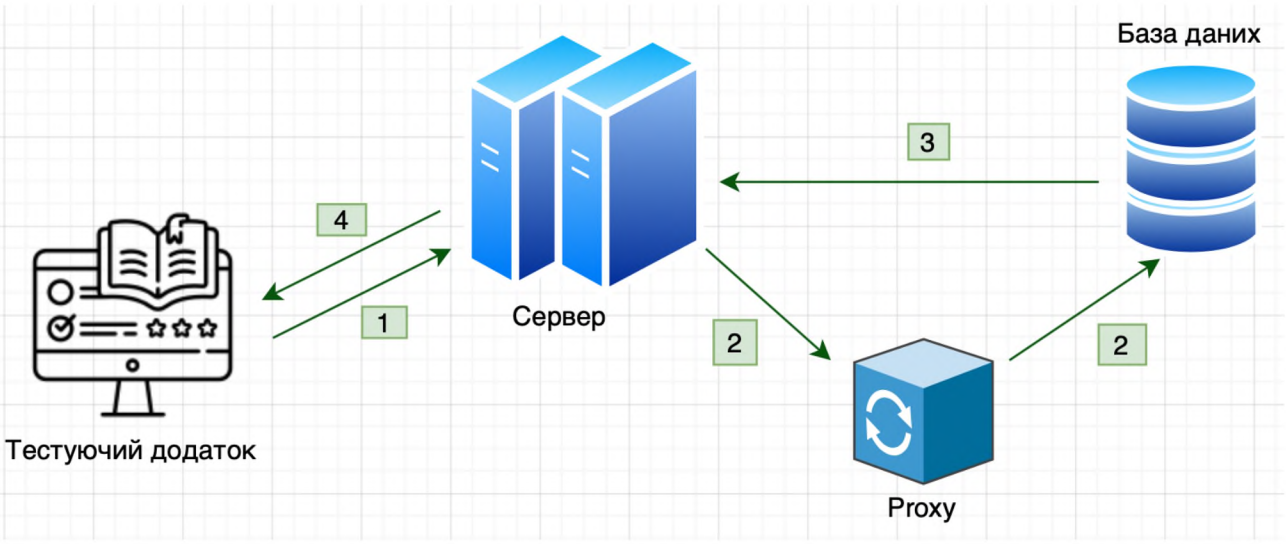


Рисунок 2 - Схема тестування

Умовні позначення:

- 1 - Тестовий запит
- 2 - SQL запит
- 3 - Відповідь бази даних
- 4 - Відповідь сайту / відображення результатів

Наступним етапом буде запуск тесту згідно з тестових сценаріїв.

На рисунку 3 зображено шаблон тестування, що коротко і чітко описує етапи, які необхідно виконати для кожного з тестового сценарію.

Вхідні дані у кожному з тестових сценаріїв будуть унікальні в залежності від типу тесту.

Для того, аби перевірити можливість реалізації SQL ін’єкції буде виконуватися додавання до очікуваного сайтом та базою даних запиту, ті символи та параметри які не очікуються і не повинні там бути присутні.

Саме тестування буде складатися з оцінки очікуваного значення параметри, потім безпосередньо додавання нових символів, що призводять до SQL ін'єкцій та оцінка отриманих результатів та відповідей від сайту.

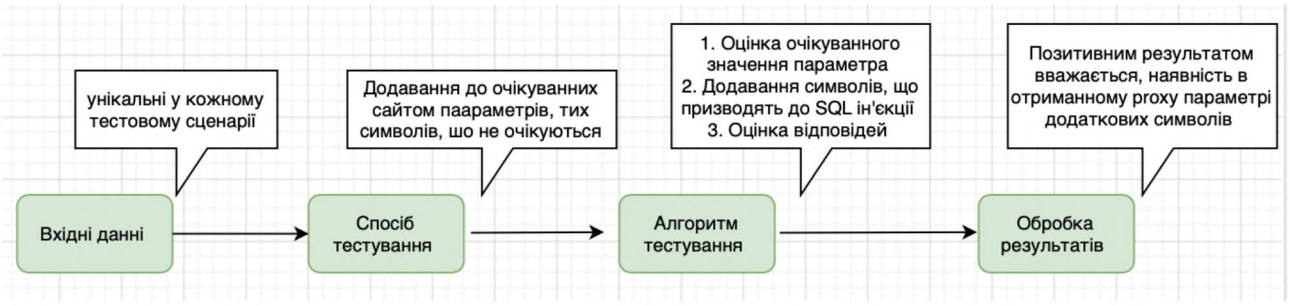


Рис 3 - Шаблон тестування

У методиці розроблено та запропоновано 5 тестових сценарій, кожен з яких призначений для перевірки основних параметрів системи.

Тестовий сценарій №1 - для визначення ступеня вразливості параметрів символів, що потрапляють на сайт;

Тестовий сценарій №2 - для визначення вразливості числових параметрів, що потрапляють на сайт;

Тестовий сценарій № 3 - для перевірки вразливостей поля пошуку, що дає можливість пошуку даних в базі даних;

Тестовий сценарій № 4 - для визначення вразливостей у полях авторизації;

Тестовий сценарій № 5 - для тестування та визначення вразливостей у формах реєстрації.

Тестовий сценарій № 1

Основне призначення сценарію визначити ступінь вразливості параметрів символів, що потрапляють на сайт.

Вхідні дані: будь-який параметр-символ.

Набір символів (додаткові символи):

1. «» - пусте значення
2. « ’ » - одинарна лапка
3. « ‘ ‘ » - дві одинарні лапки
4. « \ ‘’ » - дві одинарні лапки, перша з яких екранована символом backslash)

5. « -- » або « # » або « / * », « */ » (знак коментаря у SQL)

6. « “ » - подвійні лапки

7. « ‘ or 1`=`1 » - комбінація одинарної лапки зі правильним вираженням

8. « “ or “1”=”1 » - комбінація лапок з правильним вираженням

9. « ‘) or (‘1’= ‘1 » - комбінація одинарних лапок з вираженням

10. ***Складання тестового параметра за формулою:***

<Тестовий параметр>: = «; SELECT 0 {, 0} (зростаюча з кожною ітерацією кількість параметрів) > FROM {системні таблиці різних баз даних}».

11. ***Складання тестового параметра за формулою:***

<Тестовий параметр>: = «UNION SELECT 0 {, 0} (зростаюча з кожною ітерацією кількість параметрів) > FROM {системні таблиці різних баз даних} <знак коментаря в SQL>».

12. Вставка «/ ** /» на порожніх місцях у попередніх формулах.

13. Вставка «+» або ASCII-коду цього символу на місце прогалин в попередніх випадках.

14. Розглянуті вище набори символів з довільної комбінації літер і ASCII-кодів цих символів.

Тестовий сценарій № 2

Основне призначення сценарію це визначення вразливостей числових параметрів, що потрапляють на сайт.

Вхідні дані: будь-який чисельний параметр.

Набір символів (додаткові символи):

1. «» (пусте значення).

2. «or 1 = 1» (пробіл і справжнє вираз).

3. «-» або «#» або «/ *», «* /» (знак коментаря в SQL).

4. Розглянуті вище набори символів з довільної комбінації літер і ASCII кодів цих символів.

5. Вставка «/ ** /» на місці прогалин в попередніх випадках.

6. Складання тестового параметра аналогічно № 9 в тестовому сценарії 1.

7. Складання тестового параметра аналогічно № 10 в тестовому сценарії 1.

8. Вставка «+» або ASCII коду цього символу на місці прогалин в попередніх випадках.

9. Складання тестового параметра за формулою

<Тестовий параметр>: = <правильний параметр + 1> -1.

Тестовий сценарій № 3

Основне призначення сценарію - це перевірка вразливостей поля пошуку, що дає можливість пошуку даних в базі даних.

Вхідні дані: параметр, що передається в поле пошуку

Набір символів, що використовується для створення тестів:

1. «» (Пусте значення).

2. « '#» (Одинарна лапка і знак коментаря в SQL).

3. «%» (Знак довільного виразу при пошуку збігів оператором LIKE в SQL).

4. «% '#».

5. « '/ *» I «* /» для параметра sort (сортування), якщо є можливість встановлювати значення параметру sort.

Тестовий сценарій № 4

Основне призначення сценарію визначення вразливостей у полях авторизації.

Вхідні дані: параметри, що передаються у полях авторизації.

Спосіб тестування: додавання к імені користувача (login) та паролю (password) неочікуваних даних.

Набір символів, що використовується для створення тестів:

1. «<Ім'я існуючого користувача> 'or' 1 '=' 1» (ім'я існуючого користувача з істинним виразом).

2. «<ім'я існуючого користувача> '#» (ім'я існуючого користувача і знак коментаря в SQL).

3. Підстановка вищенаведених значень в поле пароля та комбінація цих значень в поле логіна і пароля.

4. «%» в поле пароля.

Тестовий сценарій № 5

Основне призначення сценарію це визначення вразливостей у формах реєстрації

Вхідні дані: параметри, що передаються в поля реєстрації.

Спосіб тестування: додавання до реєстраційних даних неочікуваних символів.

Алгоритм тестування: спроба авторизуватися користувачем, зареєстрованим з вірними параметрами, а потім користувачем з тестовими параметрами.

Набір символів, (додаткові символи):

1. складання тестового параметра за формулою:

<Тестовий параметр>: = {, <значення, підібрані евристичним чином>}) <знак коментаря в SQL>.

Примітка. У цій формулі закладено припущення, що проходження зберігаються параметрів в SQL-запиті, таке ж, як і порядок полів на формі. Значення в {} дужках повторюються стільки разів, скільки полів, починаючи від поля, що приймає поточний параметр, до кінця списку полів. Підбір евристичним чином означає, що тип параметрів оцінюється евристичним чином.

На цьому етапі тестування вам необхідно перевірити кожен зі створених тестових сценаріїв, дотримуючись схеми тестування на досліджуваному сайті для отримання повних та точних результатів тестування.

Заключним етапом тестування буде оцінка результатів та формування звіту з результатами тестування.

Як зазначалося вище, для визначення та правильного аналізу результатів необхідно звертати увагу на параметр тестування, іншими словами, сценарій та тест вважається позитивним, коли параметр, що був відправлений на сайт та параметр, що надійшов з сайту на proxy має у собі додаткові символи. В такому випадку можна вважати, що параметр що тестується є вразливий до атаки за допомогою SQL-ін'єкції.

Зверніть увагу, що тестування необхідно повторювати після того, як ви обновили версію Magento, після оновлення системи та сайту новим функціоналом та після встановлення додаткових розширень та налаштувань.

У методиці був розроблений шаблон звіту, який можна використовувати для презентації на аналізу результатів тестування. Шаблон звіту представлений на рисунку 4.

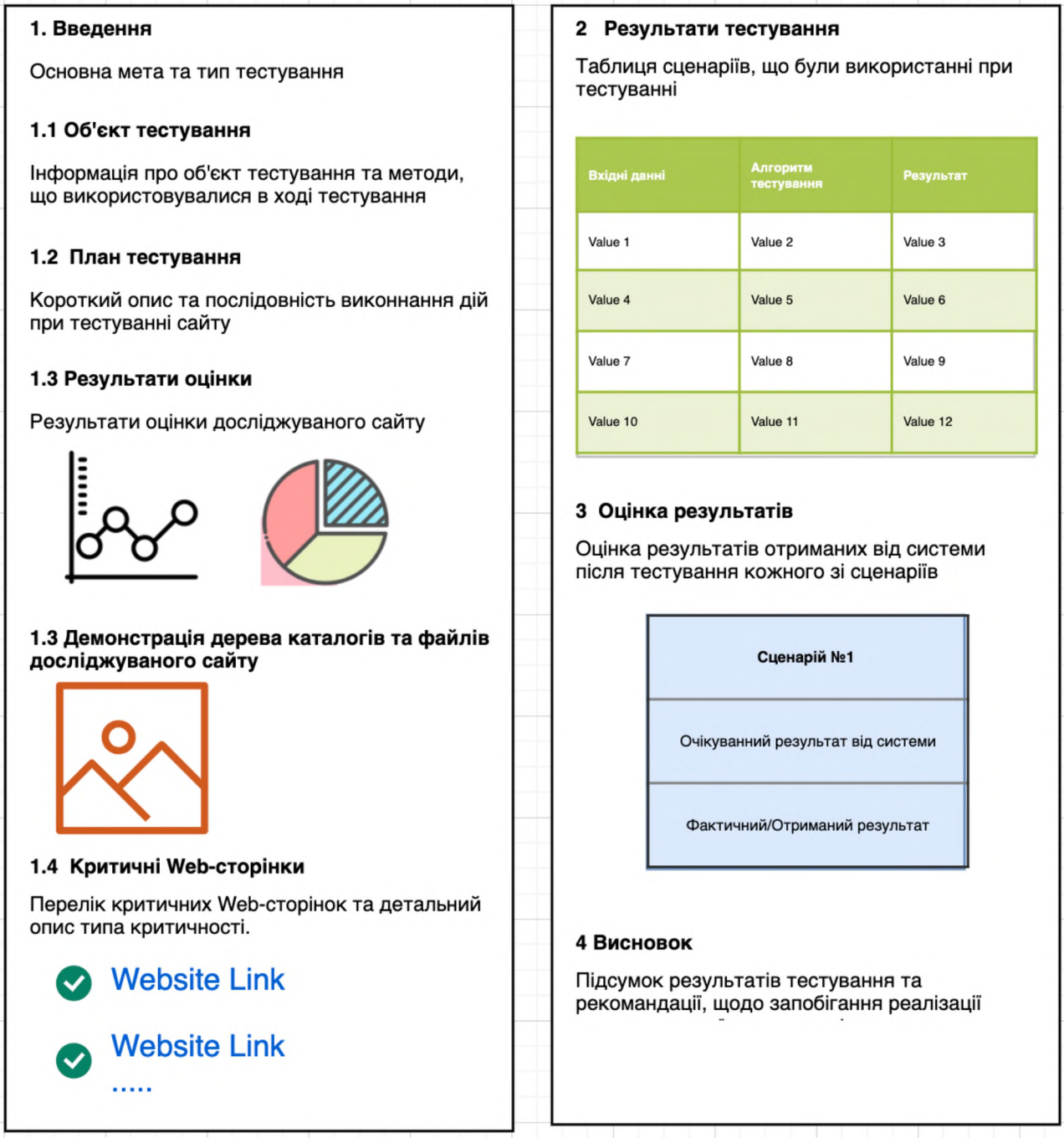


Рисунок 4 - Шаблон формування звіту

Перелік посилань

1. OWASP [Електронний ресурс] //WSTG-Latest. - 2020 – Режим доступу до ресурсу: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/05-Testing_for_SQL_Injection

2. Magento SQL injection: How to secure your Magento store against SQL-injection attack [Електронний ресурс] //ASTRA. - 2020 – Режим доступу до ресурсу: https://www.getastra.com/blog/cms/magento-security/magento-sql-injection/

3. How to test SQL injection types manually [Електронний ресурс] // Habilelabs . - 2020 – Режим доступу до ресурсу: https://www.habilelabs.io/sql-injection-types-how-to-test-sql-injection-manually/

4. Тестирование сайтов и приложений [Електронний ресурс] // GetBug . - 2018 – Режим доступу до ресурсу: http://getbug.ru/testirovanie-bezopasnosti-saytov-i-prilozheniy/

ДОДАТОК Е. Перелік документів на оптичному носії

1. Кваліфікаційна_робота_Шестерніна_125м-19-2.docx
2. Презентація_до_роботи_Шестерніна_125м-19-2.ppx

ДОДАТОК Є. Відгук керівника економічного розділу

Керівник розділу

ДОДАТОК Ж. Відгук керівника кваліфікаційної роботи

В І Д Г У К

на кваліфікаційну роботу магістра студентки групи 125м-19-2

Шестерніної Оксани Леонідівни

на тему: “Методика тестування та виявлення

SQL-ін’єкцій для веб-сайтів на CMS Magento ”

Пояснювальна записка складається зі вступу, трьох розділів і висновків, викладених на 89 сторінках.

Метою кваліфікаційної роботи є розробка методики тестування та виявлення SQL ін’єкцій для веб-сайтів на CMS Magento.

Тема кваліфікаційної роботи безпосередньо пов’язана з об’єктом діяльності магістра спеціальності 125 «Кібербезпека», а зміст та структура проекту дозволяють розкрити поставлену тему повністю.

У зв’язку з тим, що кількість кібератак на веб-сайти розроблені на CMS системах збільшується з кожним роком, а самі веб-сайти не відповідають необхідному рівню безпеки, методика тестування та виявлення вразливостей для веб-сайтів зараз дуже актуальна та необхідна для зменшення вірогідності реалізації атаки на сайт.

Практична цінність полягає у можливості використання розробленої методики для аналізу та тестуванню виявлення SQL-ін'єкцій для веб-сайтів розроблених на CMS Magento.

Оформлення пояснювальної записки до кваліфікаційної роботи виконано згідно чинних стандартів.

В ході виконання кваліфікаційної роботи студентка Шестерніна О.Л. проявила самостійність та показала добрий рівень володіння теоретичними положеннями з обраної теми. Автор уміє формулювати наукові та практичні завдання і знаходить адекватні засоби для їх вирішення та заслуговує присвоєння кваліфікації магістра за спеціальністю 125 Кібербезпека, освітньо-професійна програма «Кібербезпека» .

Рівень запозичень у кваліфікаційній роботі не перевищує вимог “Положення про систему виявлення та запобігання плагіату”.

Кваліфікаційна робота заслуговує оцінки «відмінно».

Керівник кваліфікаційної роботи

проф., д.ф.-м.н. **Т.С. Кагадій**

Керівник спец. розділу

асистент кафедри БІТ **Ю.А. Мілінчук**