МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І. С.
«____»_____ 2018 р.

ДИПЛОМНИЙ ПРОЕКТ БАКАЛАВРА
ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

«Інструментальні засоби оцінки і вибору засобів забезпечення надійності Web-сервісів**»**

Освітньо-кваліфікаційний рівень – бакалавр
Напрям підготовки: 6.050102 « Комп'ютерна інженерія»

Керівник проекту:	(підпис)	Недзельский Д. О. (ініціали, прізвище)
Консультант з охорони праці:	(підпис)	Критська Я. О. (ініціали, прізвище)
Студент:	(підпис)	Покришка С. А. (ініціали, прізвище)
Група:		КІ-14 ад

Сєвєродонецьк-2018

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

Факультетінформаційних технологій та електроніки
Кафедра комп'ютерних наукта інженерії
Напрям підготовки: 6.050102 « Комп'ютерна інженерія»
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри КІ
_____Скарга-Бандурова І. С.
"____" _____2018 р.

ЗАВДАННЯ

НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) СТУДЕНТУ

Покришка Сергій Анатолійович
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): «Інструментальні засоби оцінки і вибору засобів забезпечення надійності Web-сервісів»,

затверджена наказом по інституту від «14 » травня 2018 р. №

2. Термін подання студентом закінченого проекту (роботи):15.06.2018 р.

3. Початкові дані до проекту (роботи): матеріали переддипломної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці):

Розробити програмний комплекс для отримання цифрового електронного підпису.

Основна частина повинна містити постановку задачі, короткі теоретичні відомості, опис алгоритмів, використаних в процесі розроблення,програмне забезпечення, додатки.

5. Перелік графічного матеріалу (з точною вказівкою обов'язкових креслень): електронні плакати.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. кафедри КНІ Критська Я.О.		

8. Дата видачі завдання: 15.05.2018 р.

Керівник _____

(підпис)

Завдання прийняв до виконання

_____

(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту	Строк виконання етапів	Примітка
1.	Отримання завдання, збір матеріалів	15.05.18 – 16.05.18	
2.	Огляд літератури й обґрунтування необхідності розроблення	15.05.18– 18.05.18	
3.	Аналіз існуючих вразливостей	21.05.18 – 20.05.18	
4.	Вибір засобів для розробки	26.05.18 – 28.05.18	
5.	Розробка моделі додатку	23.04.18 – 24.05.18	
6.	Розроблення web - додатку	25.05.18 – 04.06.18	
7.	Охорона праці та безпека в надзвичайних ситуаціях	04.06.18 – 05.06.18	
8.	Оформлення пояснювальної записки	05.06.18 – 07.06.18	

Студент _____ Покришка С.А.
(підпис) (прізвище та ініціали)

Керівник _____ Недзельский Д.О.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи бакалавра: 97 с., 14 рис., 1 табл., 26 джерел, 8 додатки.

Предметом дослідницької праці є розробка додатку для аналізу та оцінки надійності web - додатків .

У вступі вибираний об’єкт дослідження, наведена його практична та теоретична важливість.

Перший розділ містить в собі загальні відомості про досліджуваний об’єкт, розкривається актуальність завдання, формулюється проблема, мета й задачі дослідження, виконується аналіз вразливостей.

В другому розділі представлена статистика вразливостей, аналіз аналогів.

В третьому розділі представлена реалізація web - додатку для сканування вразливостей та рекомендацій щодо їх усунення .

Висновки присвячені узагальненню данної роботи.

ВРАЗЛИВІСТЬ, WEB- ДОДАТОК,XSS, SQLinj, SSL

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

XSS -	Cross-site scripting
SQL inj -	structured query language injection
SSL -	secure sockets layer
LDAP -	Lightweight Directory Access Protocol
SSI -	Server-site Includes
XPath -	XML Path Language
СУБД -	Система управления базами данных
OWASP -	Open Web Application Security Project
URI -	Uniform Resource Identifier
URL -	Uniform Resource Locator

Зміст

ВСТУП 11

1 ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ВРАЗЛИВИХ МІСЦЬ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ 13

1.1 Як зловмисник бачить web-додаток 13

1.1.1 Рівень web-сервера 13

1.1.2 Рівень додатку 13

1.1.3 Рівень web-клієнта 14

1.1.4 Рівень представлення 14

1.2 Основні принципи компрометації захисту веб-додатків 15

1.2.1 Атаки на клієнт 15

1.2.1.1 Підміна вмісту 15

1.2.1.2 Міжсайтове виконання сценаріїв 16

1.2.1.3 Розщеплення HTTP-запиту 17

1.2.2 Аутентифікація 17

1.2.2.1 Підбір 17

1.2.2.2 Недостатня аутентифікація 18

1.2.2.3 Небезпечне відновлення паролів 19

1.2.3 Авторизація 19

1.2.3.1 Передбачуване значення ідентифікатора сесії 19

1.2.3.2 Недостатня авторизація 20

1.2.3.3 Відсутність таймаута сесії..... 20
1.2.3.4 Фіксація сесії 20
1.2.4 Виконання коду..... 21
1.2.4.1 Переповнення буфера..... 21
1.2.4.3 Впровадження операторів LDAP 21
1.2.4.4 Виконання команд ОС..... 22
1.2.4.5 Впровадження операторів SQL 22
1.2.4.6 Впровадження серверних розширень 23
1.2.4.7 Впровадження операторів XPath..... 23
1.2.5 Розголошення інформації..... 24
1.2.5.1 Індексування директорій..... 24
1.2.5.2 Ідентифікація додатків 24
1.2.5.3 Витік інформації 25
1.2.5.4 Дорога назад в директоріях..... 25
1.2.5.5 Передбачуване розташування ресурсів 25
1.2.6 Логічні атаки..... 25
1.2.6.1 Зловживання функціональними можливостями..... 25
1.2.6.2 Відмова в обслуговуванні 26
1.2.6.3 Недостатня протидія автоматизації 26
1.2.6.4 Недостатня перевірка процесу..... 27
1.3 Методи виявлення вразливостей в web-додатках..... 27
1.3.1 Метод отримання ідентифікуючої інформації про web-додаток 28
1.3.2 Метод тестування на проникнення 29

1.3.3 Метод статичного аналізу 29
1.3.4 Метод динамічного аналізу 30
1.4 Постановка завдання 31
Висновки 32
2 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ WEB-ДОДАТКІВ 33
2.1 Статистика вразливостей web-додатків 33
2.1.1 Аналіз предметної області 33
2.1.2 Типи атак 33
2.1.3 Установи сфери охорони здоров'я 34
2.1.4 Промислові та енергетичні компанії 35
2.1.5 Банки 36
2.1.6 IT-компанії 37
2.1.7 Державні установи 38
2.2 Способи аналізу захищеності 39
2.2.1 Інструментальний аналіз 40
2.2.2 Аналіз вручну 41
2.2.3 Аналіз вихідного коду 41
2.2.4 Комплексна оцінка 44
2.2.5 Організація процесу аналізу захищеності 45
2.3 Визначення вразливостей 46
2.3.1 SQL ін'єкція 46
2.3.2 Міжсайтовий скриптинг 47
2.3.3 Організація процесу аналізу захищеності 48
Висновок 49

3 РОЗРОБКА ТА ТЕСТУВАННЯ СВОГО WEB - ДОДАТКУ ДЛЯ АНАЛІЗУ 50
3.1 Обгрунтування вибору технологій 50
3.2 Вимоги до системи 52
3.3 Модель системи 52
3.4 Реалізація серверу 53
3.4.1 Загальні відомості 53
3.4.2 Структура бази даних 53
3.5 Розробка серверу 53
3.6 Функціонування сторінок додатку 55
3.6.1 Сторінка index.php 56
3.7 Інтерфейс сайта 56
3.8 Інструкція корстувача 59
3.9 Тестування отриманого додатку 60
Висновок 61
4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 62
4.1 Аналіз стану умов праці 62
4.1.1 Вимоги до приміщення 62
4.1.2 Вимоги до організації робочого місця 63
4.1.3 Навантаження та напруженість процесу праці 63
4.2 Виробнича санітарія 64
4.2.1 Аналіз небезпечних та шкідливих факторів при розробці виробу 64
4.2.2 Пожежна безпека 64

4.2.3 Електробезпека 65
4.3 Гігієнічні вимоги до параметрів виробничого середовища 65
4.3.1 Мікроклімат 65
4.3.2 Освітлення 66
4.3.3 Вентилювання 67
4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій 67
4.5.1 Розрахунок захисного заземлення 68
Висновки до розділу 4 72
ПЕРЕЛІК ПОСИЛАНЬ ДО РОЗДІЛУ 4 73
ВИСНОВОК 74
СПИСОК ЛІТЕРАТУРИ 75

ВСТУП

Актуальність теми дипломної роботи. Задача пошуку вразливих місць в веб-сервісах з'явилася як тільки з'явились безпосередньо самі веб-сервіси. Про актуальність цієї задачі можна судити завдяки системам оплати за вразливі місця на різних сервісах. Проблема виявлення вразливостей також являється важливою і для самих розробників веб-сервісів. Провідні гравці на ринку інформаційних технологій використовують новітні засоби утруднення експлуатації шкідливих програм і методи проактивного захисту і, тим не менш, змушені щороку виправляти сотні помилок, пов'язаних з безпекою. Зрозуміло,що для розробників краще знаходити подібні помилки в процесі аудиту сервісу, а не в результаті розслідування вже інциденту.

Прагнення виявити вразливість раніше, ніж це вдасться зловмисникам, спонукає такі компанії, як PayPal, Facebook або Google, організовувати власні програми винагороди дослідників ІБ в обмін на дані про помилки, причому бюджет найбільш великих програм перевищує мільйон доларів. Готовність сторін витрачати на інформацію про вразливість значні фінансові ресурси дозволяє припустити високу зацікавленість в інструментах, які полегшують та автоматизують процес виявлення помилок, пов'язаних з безпекою.

Метою дипломної роботи є розробка системи автоматизованого виявлення вразливостей.

Поставлена мета зумовила наступні завдання дипломної роботи:

розробити модель системи пошуку вразливостей;

імплементувати модель за допомогою відповідних програмних засобів;

розробити інтерфейс для управління системою;

оцінити ефективність отриманої системи в реальних умовах ;

Об'єктом дослідження виступає процес виявлення вразливостей в веб-сервісах.

Предметом дослідження в дипломній роботі є технології, що

застосовуються в задачі пошуку вразливостей в ПО.

Теоретична значимість дипломного дослідження полягає в аналізі існуючих методів автоматизованого виявлення вразливостей і виявлення найбільш оптимального з методів по співвідношенню складності розробки та ефективності.

Практична значимість роботи визначається можливістю повноцінного використання побудованої системи в якості одного з важливих інструментів дослідника ІБ. Прозора масштабованість і модульна структура системи при цьому дає можливість використовувати її протягом необмеженого часу.

1 ОГЛЯД ІСНУЮЧИХ ТЕХНОЛОГІЙ ВИЯВЛЕННЯ ВРАЗЛИВИХ МІСЦЬ В ПРОГРАМНОМУ ЗАБЕЗПЕЧЕННІ

1.1 Як зловмисник бачить web-додаток

1.1.1 Рівень web-сервера

До нього відноситься програмне забезпечення, що реалізує роботу програми, - це і web-сервер, і платформа виконання активного вмісту, і елементи перетворення даних для підготовки до передачі , і компоненти сторонніх систем, наприклад, коннектори систем управління даними (СУБД).

Протокол HTTP, який відповідає за транспорт всіх даних програми, у відкритому вигляді не має стану з'єднання, просто передаючи запит ресурсу та отримання відповіді клієнтом. Для зберігання даних сесії використовуються так звані cookie - область даних, що виробляється додатком і зберігається в клієнтському програмному забезпеченні. Cookie зберігаються на стороні користувача у відкритому вигляді і можуть бути йому доступні. При кожному запиті cookie передаються серверу в якості параметрів запиту. Іноді для передачі даних також можуть бути використані приховані поля.

1.1.2 Рівень додатку

На цьому рівні реалізується обробка даних програми, які були отримані від користувача, а також генерація відповідей користувачеві. Власне, тут реалізується логіка роботи програми. До цього рівня відноситься, наприклад, загальна для динамічних сайтів web-вразливість SQL ін'єкція , яка може привести до витоку всієї бази даних, використовуваної додатком. Виникає така вразливість в зв'язку з недостатньою обробкою даних, які вводить

користувач (проблема форматування і шаблонів введення), за яку повністю відповідає розробник. Якщо для користувача введення обробляється і перетворюється на стороні сервера в частину SQL-запиту, зловмисник може впровадити додаткові запити в своєму введенні і таким чином управляти віддаленою БД. Описані вище рівні відносяться до серверної складової web-додатку. На стороні користувача також можна виділити кілька логічних рівнів.

1.1.3 Рівень web-клієнта

Це прикладне програмне забезпечення (інтернет-оглядач - браузер), яке реалізує інтерфейс користувача: приймає від користувача запити, посилає їх web-серверу , відображає отримані дані , а також виконує деякий активний вміст додатку . Тут найбільша небезпека полягає в тому, що в самому браузері можуть бути вразливості. Ними може скористатися зловмисник, відправивши клієнту шкідливі дані.

1.1.4 Рівень представлення

До нього відноситься як статична, так і динамічна інформація, отримана від web-сервера і представлена в форматованому вигляді. Загрозу для цього рівня представляють атаки типу Cross-Site Scripting (XSS). Вразливості, що дозволяють реалізувати XSS, зазвичай виникають через недостатню обробку користувальницького введення, який потім відображається на web-сторінці. Вони дозволяють атакуючому впровадити код, такий як JavaScript, який виконується на стороні клієнта в web-сторінці, яку переглядає користувач. Шкідливий код, отриманий користувачем, може викрадати cookie, перенаправляти співробітника на інші web-ресурси тощо.

1.2 Основні принципи компрометації захисту веб-додатків

Під час відвідування сайту, між користувачем і сервером встановлюються довірні відносини, як в технологічному,так і в психологічному аспектах. Користувач очікує, що сайт надасть йому легітимний вміст. Крім того, користувач не очікує атак з боку сайту. Експлуатуючи цю довіру, зловмисник може використовувати різні методи для проведення атак на клієнтів сервера.

1.2.1 Атаки на клієнт

1.2.1.1 Підміна вмісту

Використовуючи цю техніку, зловмисник змушує користувача повірити, що сторінки сгенеровані Web-сервером, а не передані від зловмисника . Деякі Web-сторінки створюються динамічно. Наприклад, розташування фрейму може передаватися в параметрі URL [1]. Атакуючий може замінити значення параметра "frame_src" на своє значення . Коли буде відображатися результуюча сторінка, в рядку адреси браузера користувача відображатиметься адреса сервера, але так само на сторінці буде присутній зовнішній вміст, завантажений з сервера атакуючого, замаскований під легальний контент . Спеціально створене посилання може бути надіслане електронною поштою, опубліковане на дошці повідомлень або відкрите в браузері користувача з використанням міжсайтового виконання сценаріїв. Якщо атакуючий спровокував користувача на перехід по спеціально створеним посиланням, у користувача може скластися враження, що він переглядає дані з сервера, в той часяк частина їх була сгенерована зловмисником. Таким чином, відбудеться захист сайту на стороні

користувача,оскільки вміст сервера буде завантажено з сервера . Ця атака також може використовуватися для створення помилкових сторінок, таких як форми введення пароля, пресрелізи тощо.

1.2.1.2 Міжсайтове виконання сценаріїв

Наявність вразливості Cross-site Scripting [2][3] дозволяє атакуючому передати серверу виконуваний код, який буде перенаправлений браузеру користувача. Цей код зазвичай створюється на мовах HTML / JavaScript, але можуть бути використані VBScript, ActiveX, Java, Flash, або інші підтримувані браузером технології.

Існує два типи атак, що приводять до міжсайтового виконання сценаріїв: постійні(збережені) і непостійні (відображені). Основною відмінністю між ними є те, щоу відображеному варіанті передача коду сервера і повернення його клієнту здійснюється в рамках одного HTTP-запиту, а в збереженому - в різних.

Здійснення непостійної атаки вимагає, щоб користувач перейшов за посиланням,сформованого зловмисником (посилання може бути передано по email, ICQ і т.д.). В процесі завантаження сайту код, впроваджений в URL або заголовок запиту, буде переданий клієнту і виконаний в його браузері. Збережений різновид вразливості виникає, коли код передається серверу і зберігається на ньому деякий проміжок часу. Найбільш популярними цілями атак в цьому випадку є форуми, пошта з web-інтерфейсом і чати.

Для атаки користувачеві не обов'язково переходити по посиланню, досить відвідати вразливий сайт.

1.2.1.3 Розщеплення HTTP-запиту

При використанні даної вразливості зловмисник посилає серверу спеціальним чином сформований запит, відповідь на який інтерпретується як дві різних відповіді. Друга відповідь повністю контролюється зловмисником, що дає йому можливість підробити відповідь сервера.

У реалізації атак з розщепленням HTTP-запиту [4][5] беруть участь як правило три сторони:

- Web-сервер, що містить подібну вразливість.

- Мета атаки, що взаємодіє з Web-сервером під управлінням зловмисника. Типово в якості мети атаки виступає кешуючий сервер-посередник або кеш браузера.

- Атакуючий, який ініціює атаку.

Можливість здійснення атаки виникає, коли сервер повертає дані,надані користувачем в заголовках HTTP відповіді. Зазвичай це відбувається при перенаправленні користувача на іншу сторінку (коди HTTP 3xx) або коли дані,отримані від користувача, зберігаються в cookie.

Основою розщеплення HTTP-запиту є впровадження символів переведення рядка таким чином, щоб сформувати дві HTTP транзакції, в той час як реально буде відбуватися тільки одна. Переклад рядка використовується для того, щоб закрити першу(стандартну) транзакцію і сформувати другу пару питання/відповідь, повністю контрольовану зловмисником і абсолютно непередбачену логікою програми.

1.2.2 Аутентифікація

1.2.2.1 Підбір

Підбір [6][7] - автоматизований процес проб і помилок, що

використовується для того, щоб вгадати ім'я користувача, пароль, номер кредитної картки, ключ шифрування і т.д. Багато систем дозволяють використовувати слабкі паролі або ключі шифрування, і користувачі часто вибирають легко вгадуємі або що містяться в словниках парольні фрази. Використовуючи цю ситуацію, зловмисник може скористатися словником і спробувати використовувати тисячі або навіть мільйони комбінацій символів в якості пароля. Якщо випробуваний пароль дозволяє отримати доступ до системи, атака вважається успішною і атакуючий може використовувати обліковий запис.

Подібна техніка проб і помилок може бути використана для підбору ключів шифрування. У разі використання ключів недостатньої довжини, зловмисник може отримати використовуваний ключ, протестувавши всі можливі комбінації . Існує два види підбору: прямий і зворотній. При прямому підборі використовуються різні варіанти пароля для одного імені користувача. При зворотному - перебираються різні імена користувачів, а пароль залишається незмінним. У системах з мільйонами облікових записів ймовірність використання різними користувачами одного пароля досить висока. Не дивлячись на популярність і високу ефективність, підбір може займати кілька годин, днів або років.

1.2.2.2 Недостатня аутентифікація

Ця вразливість виникає, коли web-сервер дозволяє атакуючому отримувати доступ до важливої інформації або функцій сервера без належної аутентифікації. Інтерфейси адміністрування через web - яскравий приклад критичних систем. Залежно від специфіки додатку, подібні компоненти не повинні бути доступні без належної аутентифікації. Щоб не використовувати аутентифікацію деякі ресурси"Ховаються" за певною адресою, що не зазначена на основних сторінках сервера або інших загальнодоступних

ресурсах. Однак, подібний підхід не більше ніж "безпека через приховування". Важливо розуміти, що, не дивлячись на те, що зловмисник не знає адреси сторінки, вона все одно доступна через Web.

Необхідний URL може бути знайдений перебором типових файлів і директорій (таких як/Admin/), з використанням повідомлень про помилки, журналів перехресних посилань або шляхом простого читання документації. Подібні ресурси повинні бути захищені адекватно важливості їх вмісту і функціональних можливостей.

1.2.2.3 Небезпечне відновлення паролів

Вразливості пов'язані з недостатньою перевіркою при відновленні пароля виникають,коли атакуючий отримує можливість використовувати механізм. Це трапляється, коли інформацію, яка використовується для перевірки користувача, легко вгадати або сам процес підтвердження можна обійти. Система відновлення пароля може бути скомпрометована шляхом використання підбору, вразливостей системи або через легко вгадувані відповіді на Секретне питання.

1.2.3 Авторизація

1.2.3.1 Передбачуване значення ідентифікатора сесії

Передбачуване значення ідентифікатора сесії [10][11] дозволяє перехоплювати сесії інших користувачів. Подібні атаки виконуються шляхом передбачення або вгадування унікального ідентифікатора сесії користувача. Ця атака, також як і перехоплення сесії,в разі успіху дозволяє зловмисникові послати запит Web-сервера з правами скомпрометованого користувача.

1.2.3.2 Недостатня авторизація

Недостатня авторизація виникає, коли Web-сервер дозволяє атакуючому отримувати доступ до важливої інформації або функцій, доступ до яких повинен бути обмежений. Те, що користувач пройшов аутентифікацію, не означає, що він повинен отримати доступ до всіх функцій і вмісту сервера. Крім аутентифікації повинно бути реалізовано розмежування доступу.

Процедура авторизації визначає, які дії може здійснювати користувач, служба або додаток. Правильно побудовані правила доступу повинні обмежувати дії користувача згідно з політикою безпеки. Доступ до важливих ресурсів сайту повинен бути дозволений тільки адміністраторам.

1.2.3.3 Відсутність таймаута сесії

У разі якщо для ідентифікатора сесії [12] або облікових даних не передбачений тайм-аут або його значення занадто велике, зловмисник може скористатися старими даними для авторизації. Це підвищує вразливість сервера для атак, пов'язаних з крадіжкою ідентифікаційних даних. Оскільки протокол HTTP не передбачає контроль сесії, Web-сервери зазвичай використовують ідентифікатори сесії для визначення запитів користувача.

1.2.3.4 Фіксація сесії

Використовуючи даний клас атак [13][14], зловмисник присвоює ідентифікатору сесії користувача задане значення. Залежно від функціональних можливостей сервера,існує кілька способів "зафіксувати" значення ідентифікатора сесії. Для цього можуть використовуватися атаки типу міжсайтового виконання сценаріїв або підготовка сайту за допомогою попереднього HTTP запиту. Після фіксації значення ідентифікатора сесії

атакуючий очікує момент, коли користувач увійде в систему. Після входу користувача,зловмисник використовує ідентифікатор сесії для отримання доступу до системи від імені
користувача.

1.2.4 Виконання коду

1.2.4.1 Переповнення буфера

Переповнення буфера є найбільш поширеною проблемою в безпеці і нерідко зачіпає web-сервери. Однак атаки, що експлуатують цю вразливість,використовуються проти web-додатків не надто часто. Причина цього криється в тому, що атакуючому, як правило, необхідно проаналізувати вихідний код або образ програми.

Оскільки атакуючому доводиться експлуатувати нестандартну програму на віддаленому сервері, йому доводиться атакувати "наосліп", що знижує шанси на успіх [15].

Переповнення буфера зазвичай виникає при створенні програм на мовах C і C ++. Якщо частина сайту створена з використанням цих мов, сайт може бути вразливим для переповнення буфера.

1.2.4.3 Впровадження операторів LDAP

Атаки цього типу спрямовані на Web-сервери, що створюють запити до служби LDAP [17] на основі даних, що вводяться користувачем. Спрощений протокол доступу до служби каталогу(Lightweight Directory Access Protocol, LDAP) - відкритий протокол для створення запитів і управління службами каталогу сумісними зі стандартом X.500. Протокол LDAP працює поверх транспортних протоколів Internet (TCP / UDP). Якщо інформація, отримана

від клієнта, належним чином ніяк не перевіряється, атакуючий отримує можливість модифікувати LDAP-запит.

Запит буде виконуватися з тим же рівнем привілеїв, з яким працює компонент додатку, що виконує запит (сервер СУБД, Web-сервер і т.д). Якщо даний компонент має права на читання або модифікацію даних в структурі каталогу, зловмисник отримує ті ж можливості.

1.2.4.4 Виконання команд ОС

Атаки цього класу спрямовані на виконання команд операційної системи на Web-сервері шляхом маніпуляції вхідними даними. Якщо інформація, отримана від клієнта, належним чином ніяк не перевіряється, атакуючий отримує можливість виконати команди ОС. вони будуть виконуватися з тим же рівнем привілеїв, з яким працює компонент програми,який виконує запит (сервер СУБД, Web-сервер і т.д).

1.2.4.5 Впровадження операторів SQL

Ці атаки спрямовані на Web-сервери, що створюють SQL запити до серверів СУБД на основі даних, що вводяться користувачем.

Мова запитів Structured Query Language (SQL) є спеціалізованою мовою програмування, що дозволяє створювати запити до серверів СУБД. Більшість серверів підтримують цю мову в варіантах, стандартизованих організаціями ISO і ANSI. У більшості сучасних СУБД присутні розширення діалекту SQL, специфічні для даної реалізації (T-SQL в Microsoft SQL Server, --PL SQL в Oracle і т.д.). Багато Web-додатків використовують дані, передані користувачем, для створення динамічних Web-сторінок. З точки зору експлуатації SQL Injection [18] дуже схожий на LDAP Injection.

1.2.4.6 Впровадження серверних розширень

Атаки даного класу [19] дозволяють зловмиснику передати виконуваний код, який надалі буде виконаний на Web-сервері. Вразливості, що призводять до можливості здійснення даних атак, зазвичай полягають у відсутності перевірки даних,наданих користувачем, перед збереженням їх у інтерпретованому сервером файлі.

Перед генерацією HTML-сторінки сервер може виконувати сценарії, наприклад Server-site Includes (SSI). У деяких ситуаціях вихідний код сторінок генерується на основі даних,наданих користувачем.

Якщо атакуючий передає серверу оператори SSI, він може отримати можливість виконання команд операційної системи або включити до неї заборонений вміст при наступному відображенні.

1.2.4.7 Впровадження операторів XPath

Ці атаки спрямовані на Web-сервери, що створюють запити на мові XPath [20] на основі даних,які вводяться користувачем.

Мова XPath 1.0 розроблена для надання можливості звернення до частин документа намові XML. Він може бути використаний безпосередньо або в якості складової частини XSLT-перетворення XML-документів або виконання запитів XQuery.

Синтаксис мови XPath близький до мови SQL запитів. Припустимо, що існує документ XML,що містить елементи, відповідні іменам користувачів, кожен з яких містить три елементи - ім'я, пароль і номер рахунку. Наступний вираз на мові XPath дозволяє визначити номер рахунку користувача "dread" з паролем "1234":string (//user [name / text () = 'dread' and password / text () = '1234'] / account / text ()).

Якщо запити мови XPath генеруються під час виконання на основі призначеного для користувача введення, у атакуючого з'являється можливість модифікувати запит з метою обходу логіки роботи програми.

1.2.5 Розголошення інформації

1.2.5.1 Індексування директорій

Надання списку файлів в директорії [21] є нормальна поведінка Web-сервера, якщо сторінка, яка відображається за замовчуванням (index.html / home.html / default.htm) відсутня.

Використовуючи вміст /robots.txt або отриманого списку директорій сканер може знайти захований вміст або інші файли.

Таким чином, зовні безпечне індексування директорій може призвести до витоку важливої інформації, яка в подальшому буде використана для проведення атак на систему.

1.2.5.2 Ідентифікація додатків

Визначення версій додатків використовується зловмисником для отримання інформації про використовувані сервером і клієнтом операційні системи, Web-севери і браузери.

Також ця атака може бути спрямована на інші компоненти Web-додатків, наприклад, службу каталогу або сервер баз даних або використовувані технології програмування.

Зазвичай подібні атаки здійснюються шляхом аналізу різної інформації,наданої Web-сервером.

1.2.5.3 Витік інформації

Ці вразливості виникають в ситуаціях, коли сервер публікує важливу інформацію,наприклад коментарі розробників або повідомлення про помилки, яка може бути використана для компрометації системи. Цінні з точки зору зловмисника дані можуть міститися в коментарях HTML, повідомлення про помилки або просто бути присутніми у відкритому вигляді

1.2.5.4 Дорога назад в директоріях

Дана техніка атак [23] спрямована на отримання доступу до файлів і команд,що знаходяться поза основною директорії Web-сервера. Зловмисник може маніпулювати параметрами URL з метою отримати доступ до файлів або виконати команди, що розташовуються в файловій системі Web-сервера. Для подібних атак потенційно вразливий будь-який пристрій, який має Web-інтерфейс.

1.2.5.5 Передбачуване розташування ресурсів

Передбачуване розташування ресурсів дозволяє зловмисникові отримати доступ до прихованих даними або функціональних можливостях. Шляхом підбору зловмисник може отримати доступ до вмісту, що не призначений для публічного перегляду.

1.2.6 Логічні атаки

1.2.6.1 Зловживання функціональними можливостями

Дані атаки [24] спрямовані на використання функцій Web-додатків з

метою обходу механізмів розмежування доступу. Деякі механізми Web-додатків, включаючи функції забезпечення безпеки, можуть бути використані для цих цілей. Наявність вразливості в одному з, можливо, другорядних компонентів програми може привести до компрометації всієї програми. Рівень ризику і потенційні можливості зловмисника в разі проведення атаки дуже сильно залежать від конкретного додатка.

1.2.6.2 Відмова в обслуговуванні

Зазвичай DoS атаки спрямовані на вичерпання критичних системних ресурсів, таких як обчислювальні потужності, оперативна пам'ять, дисковий простір або пропускна здатність каналів зв'язку. Якщо якийсь із ресурсів досягне максимального завантаження,додаток цілком буде недоступний.

Атаки можуть бути спрямовані на будь-який з компонентів Web-додатку, наприклад, такі як сервер СУБД, сервер аутентифікації і т.д. На відміну від атак на мережевому рівні, які вимагають значних ресурсів зловмисника, атаки на прикладному рівні зазвичай легше реалізувати.

1.2.6.3 Недостатня протидія автоматизації

Недостатня протидія автоматизації [25] виникає, коли сервер дозволяє автоматично виконувати операції, які повинні проводитися вручну. Для деяких функцій програми необхідно реалізовувати захист від автоматичних атак. Автоматизовані програми можуть варіюватися від нешкідливих роботів пошукових систем до систем автоматизованого пошуку вразливостей і реєстрації облікових записів.

Подібні роботи генерують тисячі запитів в хвилину, що може привести до падіння продуктивності всієї програми.

Протидія автоматизації полягає в обмеженні можливостей подібних утиліт. Наприклад, файл robots може запобігати індексування деяких частин сервера, а додаткові засоби ідентифікації запобігати автоматичній реєстрації сотень облікових записів системи електронної пошти.

1.2.6.4 Недостатня перевірка процесу

Вразливості цього класу [26] виникають, коли сервер недостатньо перевіряє послідовність виконання операцій додатку. Якщо стан сесії користувача і додатку належним чином не контролюється, додаток може бути вразливим для шахрайських дій.

1.3 Методи виявлення вразливостей в web-додатках

На сьогоднішній день згідно з дослідженнями OWASP найбільш ефективним способом виявлення вразливостей web-додатків є експертний аналіз вихідних кодів web-додатків. Цей спосіб досить трудомісткий, вимагає високої кваліфікації експерта і не захищений від помилок експерта.

Тому активно розвиваються методи автоматичного виявлення вразливостей web-додатків. Методи автоматичного виявлення вразливостей web-додатків можна розділити на дві основні групи:

1) методи,які аналізують роботу розгорнутого на стенді web-додатку без звернення до вихідного коду web-додатку;

2) методи, які аналізують вихідні коди web-додатку і конфігураційні настройки.

Перша група методів розглядає web-додаток з точки зору зовнішнього користувача, тобто потенційного зловмисника. У цю групу входять такі методи:

1) Метод отримання ідентифікуючої інформації про web-додаток і виявлення його вразливостей за допомогою бюлетенів безпеки.

2) Метод тестування на проникнення.

Друга група складається з наступних методів:

3) Метод статичного аналізу вихідних кодів web-додатку.

4) Метод динамічного аналізу вихідних кодів web-додатку.

Для розробки web-додатків застосовуються різноманітні інструментальні засоби і мови програмування. Найбільш популярними серед них є PHP, Perl, Python, Ruby, Java / JSP,.Net / ASP. Можливості, що надаються цими засобами,істотно розрізняються, обмежуючи застосування тих чи інших методів аналізу. У даній роботі розглянуті методи автоматичного виявлення вразливостей і описані області застосовності та обмеження кожного з методів.

1.3.1 Метод отримання ідентифікуючої інформації про web-додаток

Метод отримання ідентифікуючої інформації про web-додаток заснований на посилці від імені звичайного користувача web-додатком набору HTTP-запитів, відповіді на які дозволять зробити висновок про те, на якому web-сервері працює додаток, за допомогою якої технології він розроблений, які версії програмного забезпечення використовує, які стандартні компоненти він використовує і т.п. Для визначення типу і версії web-сервера використовується техніка fingerprint, яка заснована на тому, що кожен web-сервер по-своєму обробляє HTTP-протокол, внаслідок чого можна з високим ступенем ймовірності визначити типі навіть версію web-сервера шляхом посилки серверу набору коректних і некоректних запитів і аналізу відповідних відповідей. Для визначення інших параметрів використовується аналіз HTTP-відповідей і HTML-сторінок засобами пошуку регулярних виразів. шаблони для пошуку задаються експертом. Наприклад,

використання розширення .jsp в URI може свідчити про те, що web-додаток було розроблено з використанням технології JSP.

1.3.2 Метод тестування на проникнення

Метод тестування на проникнення (penetration testing) розглядає web-додаток з точки зору зовнішнього користувача,тобто потенційного зловмисника. При цьому вважається, що зловмисник володіє такими ж можливостями, як і звичайний користувач, тобто не має доступу до вихідного коду web-додатку,конфігураційних налаштувань і т.п. Метод передбачає тестування працюючого на стенді web-додатку шляхом посилки запитів які емулюють призначену для користувача активність,що включає в тому числі і некоректні запити, відповідні діям зловмисника.

При пошуку вразливостей в web-додатку методом тестування на проникнення виникають три основні завдання :

• Отримання і аналіз структури web-додатку.

• Побудова набору тестових HTTP-запитів на основі побудованої структури web-додатку.

• Прогон тестового набору з аналізом відповідей web-додатку для виявлення вразливостей.

Завдання отримання і аналізу структури web-додатку полягає в тому, щоб побудувати повний список URI web-додатку,методів доступу до них і списків їх параметрів, виділити URI,захищені ідентентіфікацією. Дана інформація необхідна для побудови набору тестових запитів до web-додатку.

1.3.3 Метод статичного аналізу

Метод статичного аналізу вихідних кодів web-додатку не передбачає реального виконання. Замість цього проводиться побудова графів управління

і залежностей поданим, і виявлення вразливостей за допомогою аналізу цих графів.

Для виявлення вразливостей використовується два основні підходи:аналіз типів безпеки і аналіз потоків даних. В кожному з підходів вразливість визначається, як порушення в програмі властивості noninterference.

При аналізі типів безпеки вводиться набір типів безпеки (t1, t2,..., tn), над якими вводиться відношення часткового порядку. Кожній змінній програми ставиться в відповідність її тип безпеки. Існують два способи визначення типів безпеки всіх змінних - ручний і автоматичний. Ручний спосіб передбачає, що при кожному оголошенні змінної програміст повинен явно задати її тип безпеки. Автоматичний метод передбачає, що дані: розмітка типами безпеки змінних, що містять вхідні дані,розмітка функцій, які здійснюють для користувача введення правила виведення типів безпеки ще нерозмічених змінних. Таким чином, при аналізі програми послідовно від початку до кінця всі нерозмічені змінні отримують свій тип безпеки автоматично. Тип безпеки змінної постійний на весь час життя змінної.

Аналіз типів безпеки має істотний недолік - постійну прив'язку типу безпеки до змінної. В результаті тип безпеки визначається без урахування того, з якого джерела дані потрапили в змінну, що призводить до великої кількості помилок першого роду. Для того щоб уникнути великого числа помилок першого роду, було запропоновано прив'язувати тип безпеки не до змінної,а до її значення. В результаті кожна змінна теоретично може отримувати будь-який тип безпеки протягом свого життя. Цей підхід отримав назву аналізу потоків даних.

1.3.4 Метод динамічного аналізу

Метод динамічного аналізу вихідних кодів web-додатку за своєю суттю

аналогічний методу статичного аналізу вихідних кодів за винятком того, що аналіз проводиться в процесі виконання web-додатку без побудови графів програм. Замість цього в процесі виконання програми (тобто не явно сформований один із шляхів в графі управління) для кожної змінної проводиться обчислення типу безпеки, а для кожної розміченої функції порівнюються типи безпеки параметрів зі специфікацією, вказаною в розмітці. Метод динамічного аналізу дозволяє здійснювати виявлення вразливостей для широко застосовуваних при створенні web-додатків динамічних мов, як, наприклад, Perl, PHP, Python, Ruby. В даний час для різних мов web-додатків метод динамічного аналізу часто інтегрується в інтерпретатор мови, наприклад, Perl Tainted Mode для мови Perl,PHP revent для мови PHP і Ruby's Tainted Mode для мови Ruby.

1.4 Постановка завдання

Веб - додаток , що розробляється, має задовольняти такі потреби кінцевого користувача: сканувати сайт на вразливості, можливість обрати, які вразливості шукати, можливість отримання усіх посилань, що є на сайті, для більш детального сканування .

Інформаційна система, що проектується, має позбутися недоліків розглянутих аналогів у пункті 1.2 та мати наступні переваги:

1) безкоштовність;

2) можливість сканувати багато різних вразливостей;

3) без реєстрації.

Для досягнення поставленої задачі необхідно виконати наступне:

1) Опис та аналіз предметної області

2) Проектування бази даних

3) Розробка серверної та клієнтської частини web - додатку

Висновки

В даному розділі були розглянуті основні вразливості веб - додатків. Також було розглянуто ієрархію та рівні бачення зловмисником web - серверу .

Далі були розглянуть основні принципи компрометації веб - серверу, такі як : атака на клієнт , або виконання довільного коду.

Також були проаналізовані методи виявлення вразливостей в web - додатках.

2 ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ WEB-ДОДАТКІВ

2.1 Статистика вразливостей web-додатків

2.1.1 Аналіз предметної області

Сканери вразливостей - це програмні або апаратні засоби, що служать для здійснення діагностики та моніторингу мережевих комп'ютерів, що дозволяє сканувати мережі, комп'ютери та програми на предмет виявлення можливих проблем у системі безпеки, оцінювати і усувати уразливості.

Сканери вразливостей дозволяють перевірити різні додатки в системі на предмет наявності вразливостей, якими можуть скористатися зловмисники.

2.1.2 Типи атак

У 2017 року найбільш поширеною була атака «SQL ін'єкція»; в разі її успішної реалізації зловмисник може отримати несанкціонований доступ до чутливої інформації або виконати команди ОС. На другому місці рейтингу атака на користувачів веб-додатків «міжсайтове виконання сценаріїв». Ці типи атак як і раніше становлять практично половину від усіх атак на досліджувані веб-додатки. Виросла частка атак «Підключення локальних файлів», спрямованих на виконання довільного коду на атакується сервері. Даний тип атак посідає третє місце в рейтингу. Крім того збільшилася кількість атак високого ступеня ризику «Віддалене виконання коду і команд ОС», за допомогою яких зловмисник може отримати повний контроль над сервером з веб-додатком.

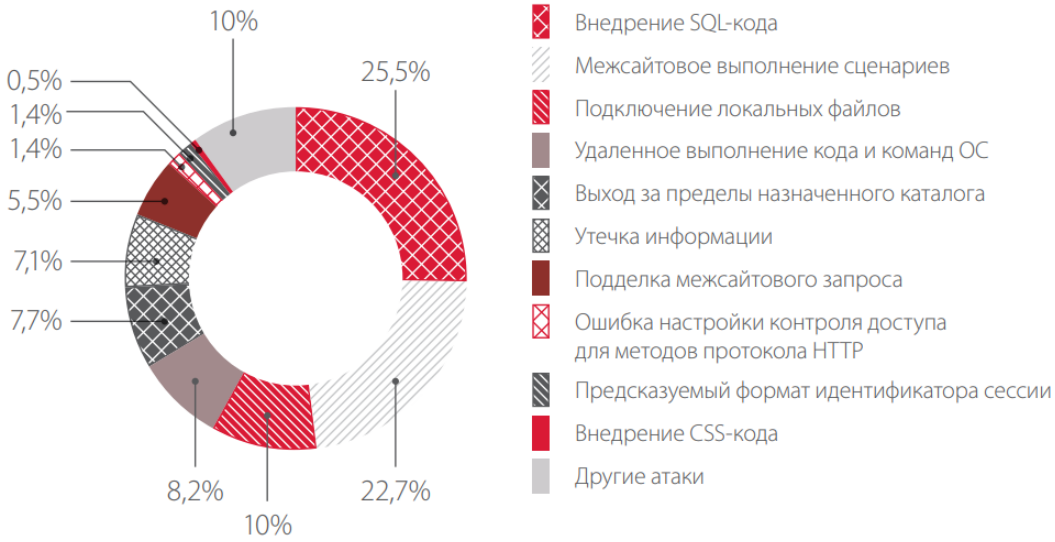


Рисунок 2.1 - Типи атак на веб-додатки

2.1.3 Установи сфери охорони здоров'я

Статистика по найпоширенішим атакам на веб-додатки сфери охорони здоров'я в порівнянні з минулим кварталом значно відрізняється. Ця відмінність пояснюється специфікою роботи цих додатків. В основному досліджені в цьому кварталі веб-додатки використовуються в якості інформаційних ресурсів і не використовуються для обробки персональних даних або відомостей про стан здоров'я пацієнтів.

Саме відсутністю чутливої інформації, яка може представляти інтерес для зловмисників, пояснюється значне зниження частки атак «SQL ін'єкція» (з 46% до 2,9%) у порівнянні з минулим кварталом. При цьому зросла кількість інших атак, серед яких практично половина спрямована на виконання коду або команд ОС, в тому числі за допомогою підключення довільних файлів. За допомогою подібних атак зловмисник може отримати повний контроль над веб-додатком і потім підміняти вміст ресурсу, порушувати його роботу або поширювати шкідливе ПЗ.

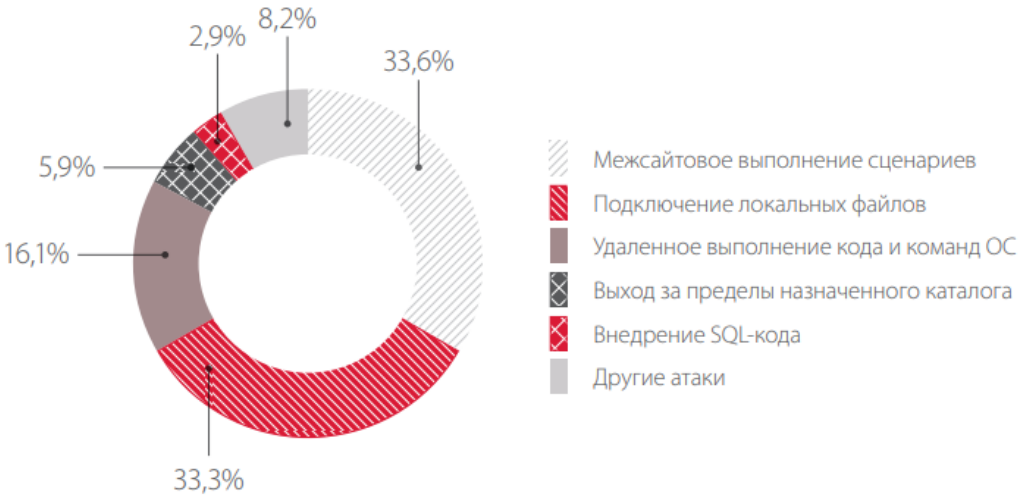


Рисунок 2.2 - Типи атак на веб - додаток сфери охорони здоров'я

2.1.4 Промислові та енергетичні компанії

Атаки на веб-додатки промислових і енергетичних компаній як і раніше носять цілеспрямований характер. Зловмисники, атакуючи подібні веб-додатки, намагаються отримати чутливу інформацію про систему, яка може бути використана при плануванні і проведенні подальших атак. Як ми відзначали в дослідженнях першого півріччя, зловмисники намагаються використати такі веб-додатки в якості основної точки для проникнення у внутрішню інфраструктуру компаній з подальшим доступом до технологічних сегментах мережі. Кінцевою метою подібних атак в основному є порушення технологічного процесу. Тому серед найпоширеніших атак присутні ті, які дозволяють віддалено виконати команди ОС і захопити контроль над сервером і надалі розвивати вектор атаки на внутрішню інфраструктуру.

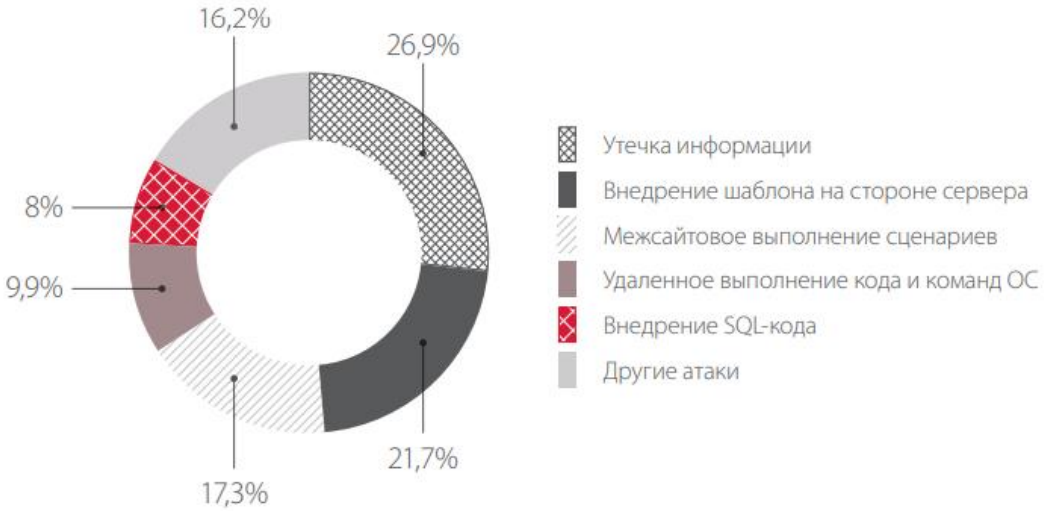


Рисунок 2.3 - Типи атак на веб - додатки промислових та енергетичних компаній

2.1.5 Банки

Отримання фінансової вигоди є основним мотивом зловмисників при проведенні атак на веб-ресурси банків. За допомогою атак «SQL ін'єкція» порушники можуть дістати несанкціонований доступ до чутливої інформації, в тому числі до персональних даних та фінансовим відомостями клієнтів банків. Крім того, практично кожна третя атака спрямована на користувачів веб-додатків. Зловмисники намагаються виявити уразливості, за допомогою експлуатації яких вони зможуть викрадати облікові дані користувачів або заражати їх робочі станції шкідливим ПЗ. Успішна реалізації таких атак може привести до фінансових втрат як для клієнтів, так і для самого банку, а також негативно вплинути на його репутацію.

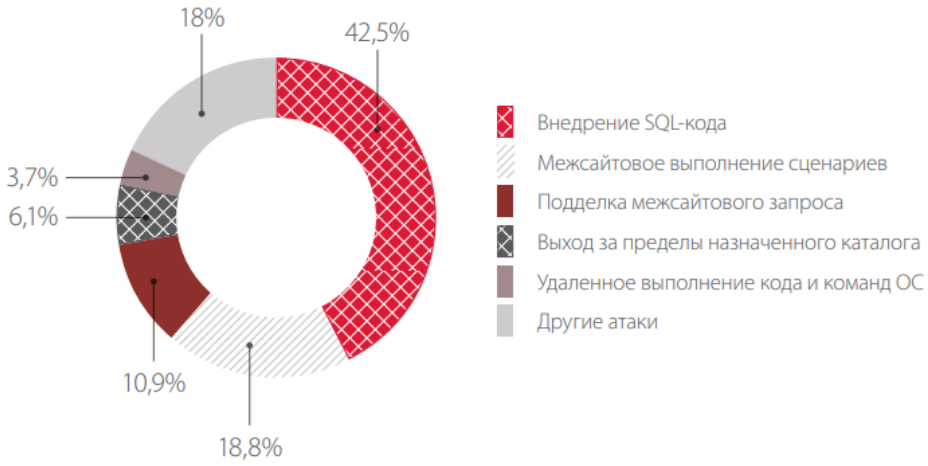


Рисунок 2.4 - Типи атак на веб - додатки банків

2.1.6 IT-компанії

Практично половину всіх атак складають «SQL ін'єкція» та «міжсайтовий виконання сценаріїв». Дані атаки в першу чергу спрямовані на доступ до чутливої інформації та на отримання облікових записів користувачів веб-додатків з метою подальшого доступу до інформаційних ресурсів організації. Наприклад, в разі успішної атаки «міжсайтовий виконання сценаріїв» зловмисник може отримати cookie користувача та здійснити доступ до порталу, який компанія використовує для взаємодії з партнерами. Часто на таких ресурсах обробляється чутлива інформація, яка представляє високу цінність для зловмисників. Крім того, особливий інтерес для зловмисників представляють облікові дані привілейованих користувачів, які володіють правами адміністратора і часто використовують одну і ту ж обліковий запис для доступу до декількох веб-додатків і порталів. Тому атаки «SQL ін'єкція» і «міжсайтовий виконання сценаріїв» можуть бути спрямовані саме на отримання облікових даних адміністраторів веб-додатків. В результаті доступ порушника до партнерських порталам і іншим веб-додатків може принести IT-компанії істотні репутаційні та, можливо, фінансові втрати.

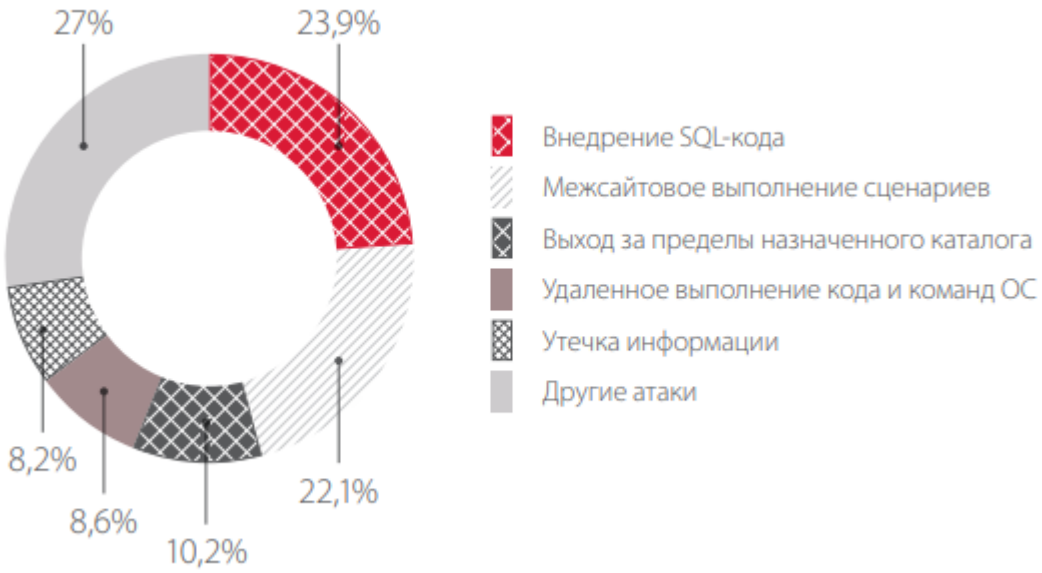


Рисунок 2.5 - Типи атак на веб - додатки IT-компаній

2.1.7 Державні установи

Велика частина веб-додатків державних установ, що розглядаються в рамках даного дослідження, використовується для обробки персональних даних громадян або застосовується в якості інформаційних і новинних ресурсів. Кожна п'ята атака зловмисників направлена на несанкціонований доступ до чутливої інформації, кожна друга - на користувачів веб-додатків . Зловмисники користуються тим, що більшість користувачів таких веб-ресурсів дуже погано інформовані в питаннях інформаційної безпеки. Основна мета порушників - отримання персональних даних користувачів.

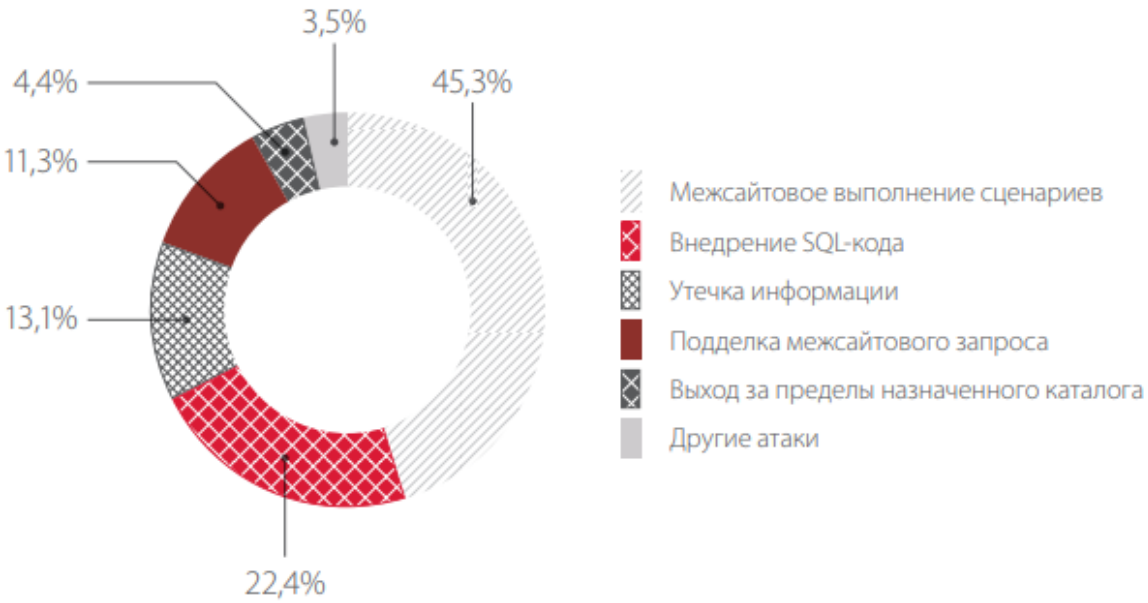


Рисунок 2.6 - Типи атак на веб - додатки державних установ

2.2 Способи аналізу захищеності

Сьогодні є велика кількість матеріалів, присвячених проведенню аналізу захищеності Web-додатків. Найбільш відома - бібліотека документів Open Web Application Security Project (OWASP), що містить повне керівництво з пошуку всіляких вразливостей, їх класифікацію, а також рекомендації до процесу проведення аналізу захищеності.

З точки зору класифікації вразливостей цікаві матеріали проекту Web Application Security Consortium (WASC). Варто також відзначити відкритий стандарт Open Source Security Testing Methodology Manual (OSSTMM), який розглядає аналіз захищеності Web-додатків в контексті комплексного процесу проведення тестування на проникнення.

2.2.1 Інструментальний аналіз

Даний спосіб в першу чергу передбачає використання сканерів безпеки, а також додаткових інструментів, що автоматизують деякі сценарії експлуатації та виявлення вразливостей.

Інструментальне обстеження є найбільш простим і як наслідок найбільш поширеним способом аналізу захищеності Web-додатків. За простоту доводиться розплачуватися помилками «другого роду» (false negative) - пропуском частини вразливостей, яким піддається досліджуваний додаток. Наприклад, сьогодні сканери не можуть самостійно (тобто без використання сигнатур) виявити такі уразливості, як небезпечне відновлення паролів (Weak Password Recovery Validation), відсутність тайм-ауту сесії (Insufficient Session Expiration) або логічні атаки.

Якщо об'єктом атаки буде додаток, що використовує робочий процес, в свою чергу реалізує якусь бізнес-завдання, то це додаток може виявитися уразливим, якщо не здійснюються перевірки виконання всіх попередніх етапів процесу. Припустимо, що існує Web-додаток, що використовує певний процес для активації грошової суми з пластикової карти. На першому етапі цього процесу додаток засвідчується в коректності даних, що вводяться користувачем, потім йде перевірка, числиться ця пластикова карта в базі клієнтів банку і чи відповідає запитувана сума номіналу рахунку. При негативному результаті користувач отримує повідомлення про помилку. На останньому етапі цього процесу відбувається зміна балансу рахунку користувача відповідно до введених даними на першому етапі. Успішна атака на такий додаток може бути реалізована в тому випадку, якщо існує можливість звернутися до останнього етапу процесу (поповнення рахунку) з передачею довільної суми для зміни балансу. Сканер безпеки пропустить подібну уразливість - йому не відомо, що була виконана операція поповнення рахунку з довільною сумою. Все, що буде мати у своєму розпорядженні

сканер безпеки, це те, що при зверненні до «такий-то» сторінці, на ній змінюється «таке-то» значення і з його точки зору - це не є вразливістю. За таким же принципом може працювати цілком безпечне додаток, наприклад новинна стрічка або дошка оголошень.

Варто відзначити, що, незважаючи на деякі обмеження інструментального аналізу захищеності, даний спосіб задовольняє вимогам стандарту Payment Card Industry Data Security Standard (PCI DSS) при проведенні оцінки захищеності вузлів, доступних в Internet.

2.2.2 Аналіз вручну

Пошук у ручному режимі в більшості випадків дозволяє виявити уразливості, які неможливо виявити інструментальним шляхом, проте вимагає більше часу. При цьому якість аналізу сильно залежить від рівня знань фахівця в даній галузі і його досвіду проведення подібних робіт. Важливо розуміти, що коли більшість перевірок здійснюється вручну, то зростає ризик прояву «людського фактора». Однак існують Web-додатки, щодо яких неможливо або вкрай важко провести інструментальне обстеження, такі як програми з банківської сфери, що використовують модель захисту від вразливостей, заснованих на так званій «межсайтовой підробці запитів» (Cross-Site Request Forgery, CSRF). В цьому випадку єдиним способом аналізу захищеності є виконання всіх перевірок вручну.

2.2.3 Аналіз вихідного коду

На відміну від аналізу вручну, аналіз вихідного коду дозволяє перевірити захищеність Web-додатки, не зачіпаючи його роботу, - дослідження може виконуватися без доступу до самого Web-сервера, якщо не потрібне виконання перевірок експлуатації уразливості. Це найбільш

безпечний спосіб проведення подібних робіт. Одним з недоліків цього способу є неможливість провести оцінку захищеності стану Web-ресурсу - тут потрібно безпосередній доступ до нього.

Аналіз вихідного коду дозволяє виявити багато уразливості, яким піддається Web-додаток, однак для цього може знадобитися досить багато часу, яке залежить від складності стилю програмування, що використовується при написанні програми, і обсягу аналізованого коду. Крім того, існує безліч комерційних додатків, розробники яких поширюють свої продукти тільки у вигляді бінарних файлів (наприклад, з використанням кодування Zend або технології ASP.NET), що не дозволяє застосувати цей спосіб. Хоча варто зауважити, що можна проводити і бінарний аналіз додатки (цим, зокрема, займається компанія Veracode).

На практиці аналіз вихідного коду здійснюється в повному обсязі переважно лише для окремих модулів або функцій обстежуваного додатки. У разі коли необхідний аналіз вихідного коду на наявність вразливостей всього програми, вдаються до автоматизованого способу, проведеного статично або динамічно.

Пошук методом статистичного аналізу вихідного коду заснований на використанні сигнатур, які в свою чергу базуються на апараті регулярних виразів. Даний метод в силу своєї простоти набув найбільшого поширення. Необхідно враховувати, що при використанні статичного аналізу неминучі помилки першого і другого роду, коли аналізує додаток не зможе виявити деяке число вразливостей в силу відсутності відповідної сигнатури. Можливі також численні помилкові повідомлення про наявність уразливості.

Помилки першого роду (false positive - «незнайдені уразливості») при використанні статичного аналізу вихідного виникають в силу наступних причин:

1) при програмуванні Web використовується складний синтаксис;

2) перевірки змінних відбуваються з використанням власних функцій

програми;

3) відсутні відповідні сигнатури.

Динамічний аналіз більш ефективний у порівнянні зі статичним. Засіб, що дозволяє виконати динамічний аналіз коду, досконально розбирає синтаксис мови програмування, на якому написано досліджуваний додаток, і проводить ряд перевірок, спрямованих на виявлення необроблених даних з боку користувача, що надходять в потенційно небезпечні функції програми в якості аргументів. Такий аналіз дозволяє за короткий час виявити всі грубі помилки, допущені програмістами, і видати звіт з мінімальною кількістю помилок першого і другого роду. Однак через складність реалізації таких засобів, а також в силу необхідності підтримки безлічі мов програмування (PHP, ASP, Perl, Python, Java та ін.), На яких можуть бути написані Web-додатки, подібних засобів не так вже й багато. Найбільшого поширення набули Coverity, Valgrind і Fortify PTA.

Варто враховувати, що автоматизованим аналізатора вихідного коду притаманні ті ж недоліки, що і сканерів безпеки. Наприклад, неможливо виявити вразливості «Небезпечне відновлення паролів», «Відсутність тайм-ауту сесії», «Логічні атаки» тощо. Тому, як правило, при всебічному аналізі вихідного коду доводиться комбінувати використання одного або обох методів автоматизованого аналізу вихідних текстів, а також проводити експертний аналіз окремих модулів програми: аутентифікації, авторизації, відновлення паролів, проводки транзакцій і ін.

З найбільш відомих розробників комерційних продуктів з автоматизованого аналізу вихідних текстів Web-додатків можна виділити компанії Armorize Technologies, Fortify і Ounce Labs.

2.2.4 Комплексна оцінка

Даний спосіб дозволяє провести аналіз захищеності Web-додатки з позицій середовища його функціонування, що буває корисно в разі сервіс-орієнтованих архітектур і при проведенні аудиту інформаційної системи в цілому. По суті, можна по-різному комбінувати перераховані підходи до аналізу захищеності, доповнюючи процес дослідження такими можливостями, як наприклад Compliance (оцінка відповідності певним критеріям). При проведенні аналізу можуть використовуватися такі принципи.

Принцип «чорного ящика» (black-box). Проведення робіт з оцінки захищеності додатка без попереднього отримання будь-якої інформації про нього. Це корисно, коли необхідно оцінити захищеність з позицій зловмисника, зазвичай має в своєму розпорядженні мінімальними знаннями про досліджувану систему. В основному подібні оцінки здійснюються в рамках «тестування на проникнення» (penetration testing). Всі дослідження можуть проходити як з попередженням обслуговуючого персоналу про плановані роботах, так і без нього. У другому випадку існує можливість оцінити, за який час після початку дослідження персонал зафіксує інцидент, а також яка адекватність дій, що робляться по мінімізації його впливу або запобігання.

Принцип «сірого ящика» (gray-box). Проведення робіт з наданням всієї необхідної інформації про програму, крім забезпечення безпосереднього доступу до самого сервера, на якому функціонує досліджуване Web-додаток. Зазвичай виконавцю надаються такі дані: структура каталогів додатки, дані для авторизованого підключення в просторі Web-додатки (наприклад, ім'я користувача, пароль і набір одноразових паролів для проводки транзакцій), вихідний код деяких файлів або функцій та ін.

Принцип «білого ящика» (white-box). Даний принцип має на увазі

передачу виконавцю всього програми з його подальшим розгортанням на майданчику консультанта, залученого до робіт по його аналізу, або організацію аналогічної копії додатка у власній інформаційній системі з наданням виконавцю повного доступу до цього ресурсу. В даному випадку є можливість відстежити, яким чином додаток реагує на будь-який передається до нього запит. Це найбільш продуктивний метод проведення аналізу захищеності Web-додатків, що дозволяє виявити найбільше число вразливостей. Однак варто зауважити, що даний метод не має змоги поглянути на додаток з позицій атакуючого.

2.2.5 Організація процесу аналізу захищеності

Перш за все, необхідно зрозуміти, яку мету має переслідувати аналіз, а потім визначити область дослідження і, керуючись стратегією управління інформаційними ризиками і допустимим бюджетом, сформувати перелік перевірок. Якщо мета аналізу полягає в демонстрації можливості проникнення, порушення штатного режиму роботи програми або демонстрації компрометації чутливої інформації, тоді роботи варто організувати за принципом «чорного ящика» без обмежень по проведеним перевіркам.

Результати подібного дослідження продемонструють поточний стан справ з безпекою об'єктів дослідження. У разі низького рівня захищеності Web-додатків подібні роботи наочно демонструють реалізовані загрози з боку зовнішнього порушника, і наслідком цього може стати виділення додаткового бюджету на роботи з мінімізації ризиків.

Коли мета аналізу захищеності полягає в підвищенні рівня безпеки програми в умовах обмеженого бюджету, то краще за все організувати процес аналізу ресурсу методом «сірого ящика» з використанням інструментального підходу до його обстеження з частковими перевірками,

виконаними вручну.

Який би варіант проведення обстеження не був обраний, важливо створити резервні копії ресурсу до початку проведення.

2.3 Визначення вразливостей

Для того, щоб автоматизувати пошук вразливостей спочатку потрібно зрозуміти як взагалі можна знаходити вразливості.

2.3.1 SQL ін'єкція

SQL-ін'єкція є виконання довільного запиту до бази даних програми за допомогою поля форми або параметра URL. У разі використання стандартного мови Transact SQL можливо вставити шкідливий код. В результаті чого будуть отримані, змінені або видалені дані таблиць. Щоб запобігти цьому, потрібно використовувати параметризрвані запити, які підтримуються більшістю мов веб-програмування.

Розглянемо запит:

```
SELECT * FROM table WHERE column = 'parameter';
```

Якщо зловмисник змінить значення parameter на 'OR' 1 '=' 1, запит прийме наступний вигляд:

```
SELECT * FROM table WHERE column = '' OR '1'='1';
```

Так як '1' дорівнює '1', атакуючий отримає доступ до всіх даних таблиці. Це дозволить виконати довільний запит, додавши в кінець виразу SQL.

Уразливість цього запиту легко усунути за допомогою параметризації. Наприклад, для програми, написаної з використанням PHP і MySQLi, він виглядає так:

```
$stmt = $pdo->prepare('SELECT * FROM table WHERE column = :value');
$stmt->execute(array('value' => $parameter));
```

2.3.2 Міжсайтовий скриптинг

Міжсайтовий скриптинг (XSS) - тип атаки на веб-ресурси, що полягає у впровадженні в сторінку сайту шкідливого коду, який виконується на комп'ютері користувача, змінює сторінку і передає вкрадену інформацію зловмисникові.

Наприклад, якщо на сторінці коментарів немає перевірки вхідних даних, зловмисник впроваджує шкідливий код JavaScript. В результаті у користувачів, які переглядають коментар, виконується код, і дані про авторизацію з cookies-файлів відправляються атакуючому.

Особливо схильні до цього виду атаки сучасні веб-додатки, де сторінки побудовані з користувацького контенту, що інтерпретується фронтенд-фреймворками начебто Angular і Ember. У ці фреймворки вбудовано захист від міжсайтового скриптинга, але змішане формування контенту на стороні сервера і клієнта створює нові комплексні атаки: впровадження директив Angular або хелперів Ember.

При перевірці зосередьтеся на призначеному для користувача контент, щоб уникнути некоректної інтерпретації браузером. Це схоже на захист від SQL-ін'єкцій. При динамічної генерації HTML-коду користуйтеся спеціальними функціями для зміни і отримання значень атрибутів (наприклад, element.setAttribute і element.textContent), а також шаблонизатор, які виконують екранізацію спеціальних символів автоматично.

Політика безпеки вмісту (CSP) - ще один інструмент захисту від XSS-атак. CSP - заголовки сервера, що визначають білий список джерел, звідки дозволена завантаження даних для різних типів ресурсів. Наприклад, заборона запуску скриптів зі стороннього домену та вимкнення системи eval

(). Завдяки політикам CSP навіть при впровадженні шкідливого коду в сторінку його виконання стає неможливим. На офіційному сайті Mozilla розміщено керівництво по CSP з прикладами конфігурацій.

2.3.3 Організація процесу аналізу захищеності

Перш за все, необхідно зрозуміти, яку мету має переслідувати аналіз, а потім визначити область дослідження і, керуючись стратегією управління інформаційними ризиками і допустимим бюджетом, сформувати перелік перевірок. Якщо мета аналізу полягає в демонстрації можливості проникнення, порушення штатного режиму роботи програми або демонстрації компрометації чутливої інформації, тоді роботи варто організувати за принципом «чорного ящика» без обмежень по проведеним перевіркам.

Результати подібного дослідження продемонструють поточний стан справ з безпекою об'єктів дослідження. У разі низького рівня захищеності Web-додатків подібні роботи наочно демонструють реалізовані загрози з боку зовнішнього порушника, і наслідком цього може стати виділення додаткового бюджету на роботи з мінімізації ризиків.

Коли мета аналізу захищеності полягає в підвищенні рівня безпеки програми в умовах обмеженого бюджету, то краще за все організувати процес аналізу ресурсу методом «сірого ящика» з використанням інструментального підходу до його обстеження з частковими перевірками, виконаними вручну.

Який би варіант проведення обстеження не був обраний, важливо створити резервні копії ресурсу до початку проведення.

Висновок

В цьому розділі було проаналізовано вразливості та практичні методи їх знаходження . Також було отримано статистику вразливостей сайтів різних сфер . Було наглядно показано які вразливості попадаються частіше в тих чи інших сферах діяльності.

Було розглянуто методи оцінки захищеності web - додатків.

В кінці глави було порівняння аналогічних сервісів для сканування додатків на вразливості та виділені їх недоліки.

3 РОЗРОБКА ТА ТЕСТУВАННЯ СВОГО WEB - ДОДАТКУ ДЛЯ АНАЛІЗУ

3.1 Обгрунтування вибору технологій

У попередньому розділі було дано аналіз існуючих методів виявлення вразливостей. Для реалізації в рамках атестаційної роботи було вирішено розробляти web - додаток . Основним аргументом на користь цього вибору послужило те, що web - додаток буде легкий в користуванні та доступний з будь-якого гаджету, що дозволить сканувати web - сервіси навіть с телефона.

Додатковим аргументом стало те, що нема точного аналогу web - додатку, що розробляється. Також в таблиці 3.1 порівняння додатків

	Desktop додаток	Web - додаток
Установка / оновлення	Повинно бути розгорнуто або встановлено.	Одноразова настройка. Одна установка для всіх користувачів. Завдяки централізованності моментально оновлення.
Інтерфейс взаємодії	Стандартні інтерфейси, стандартне взаємодія	Різноманітний інтерфейс взаємодії. Плюси - різноманітність реалізації, мінуси, складності - кросбраузерності сумісність. Вирішується застосуванням бібліотек на JavaScritp, впровадженням стандартів.
Сумісність з пристроями	Залежність від платформи. Виняток - Кросплатформені	У більшості випадком - від платформи незалежне.

	додатки.	
Анімація, графіка	Швидка, швидкий відгук	Відносне повільний відгук, пов'язаний з передачею даних по мережі.
Пошук по контенту	Ні, якщо тільки не реалізовано на рівні додатку.	Так є. Причому можна орагнізовать свій пошук, але і скористатися сторонніми сервісами, наприклад запитувати данни у Google.
Розповсюдження	Якщо тільки додатково налаштувати	Спочатку веб-додатки (більшість) налаштовані на спільний доступ
Розробка	Під кожну платформу є свої інструменти, найчастіше під кожну платформу доводитися писати свою версію.	Все виконується на сервері, користувача не хвилює як там виповнюється все на сервері. Кроссплаформенно, потрібен тільки браузер. Інструменти, софт на сервері часто кроссплатформенную.
Тестування	Проводиться тестером, групою тестерів. Для opensource відбувається тестування усіма, кому це цікаво.	По суті все так же. Тільки відкритість (розташування в мережі) даного роду додатків дозволяє залучити більша кількість тестерів. Сотні, тисячі, мільйони. В результаті більше покриття тестами і більш швидке виявлення вразливостей і некоректної роботи софта.

Таблиця 3.1 - порівняння desktop та web додатків

3.2 Вимоги до системи

Вибір технології тестування визначив такі основні вимоги до системи:

1) Автоматизації системи.

2) Централізоване управління.

3) Збереження результатів аналізів

4) Надання рекомендацій щодо забезпечення надійності web - додатку

3.3 Модель системи

Набір ключових компонентів і їх функцій був визначений виходячи з пред'явлених до системи вимог. Ознайомитися з іехрархіей моделі можна на малюнку 3.1.

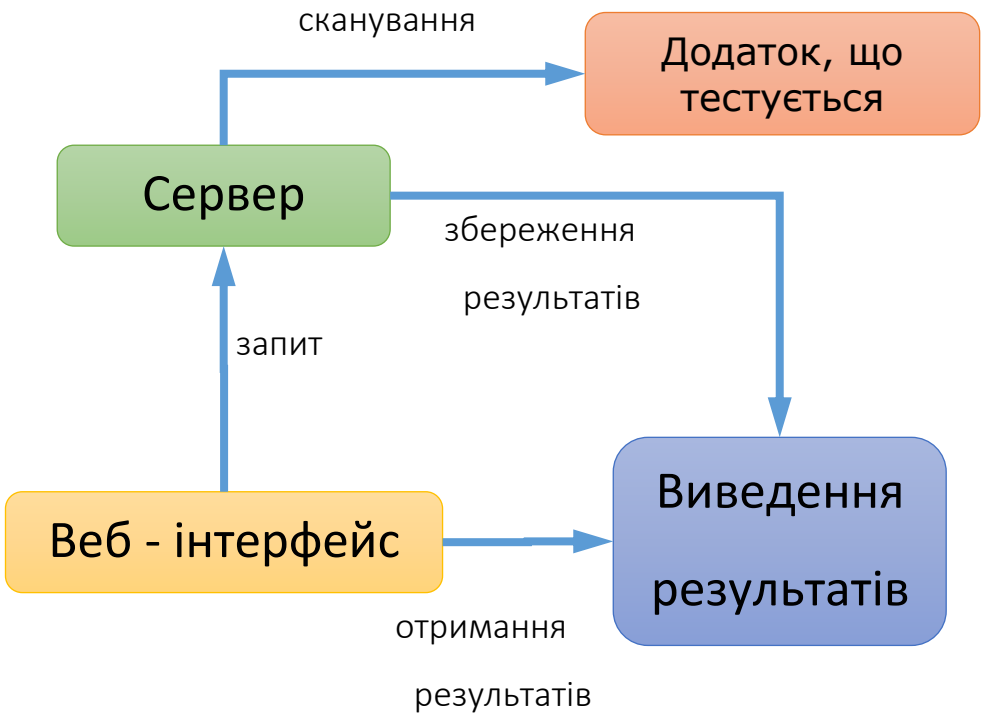


Рисунок 3.1 - модель системи веб - додатку

Далі кожен компонент окремо.

3.4 Реалізація серверу

3.4.1 Загальні відомості

Для реалізації серверної частини було обрано мову програмування php, базу даних MySQL та було використано бібліотеки httpclient та PHPCrawl_071.

Бібліотека httpclient слугує для полегшення роботи з віддаленими серверами, та обробки інформації отриманої від них.

Бібліотека PHPCrawl_071 слугує для отримання всіх посилань чи активних скриптів на сторінці для подальшої обробки та сканування їх на вразливості.

3.4.2 Структура бази даних

База даних містить три таблиці : tests,tests_result та vulnerabilities.

Опис таблиць:

tests містить інформацію про всі сканування;

tests_result містить інформацію про всі знайдені вразливості ;

vulnerabilities містить інформацію про вразливості та способу їх усунення. Для кожної вразливості, яку може виявити сканер , існує свій запис.

3.5 Розробка серверу

Основний функціонал сайту знаходиться в папці tests, де знаходяться файли з функціями для знаходження вразливостей .

Функція testForStoredXSS відповідає за знаходження збережених xss. При проведенні «базової відображеної» XSS-атаки не мають на меті крадіжки конфіденційної інформації. Суть даної комп'ютерної атаки полягає у тому, що при переході на веб-сайт з «відбитою» XSS-вразливістю, виводиться застережливе вікно. Це результат виконання скриптового коду. Посилання для переходу на веб-сторінку з «базової відображеної» XSS-вразливістю виглядає приблизно так:

```
http://site.ua/<script>alert("XSS")</script>
```

Функція testAutoComplete відповідає за знаходження полів з авто заповненням. Додаток сканує всі знайдені сторінки на наявність на них форм та полів для введення. Після чого просто перевіряється наявність атрибуту autocomplete в кожному полі та зберугаються результати, якщо вразливість була знайдена.

Функція testUnvalidatedRedirects знаходить неправильні редіректи.

Як і всі попередні функції вона відправляє на кожну знайдену сторінку запит з доменом, якщо після цього домен змінюється на той, що був вказаний - то вразливість знайдена.

Функція testHttpBannerDisclosure знаходить доступну інформацію про сервер.

Функція testForReflectedXSS знаходить відображені XSS.

Даний тип XSS-атаки отримав таку назву, оскільки реалізується через DOM (Document Object Model), що не залежить від платформи і мови програмний інтерфейс, що дозволяє програмам і сценаріям отримувати доступ до вмісту HTML і XML-документів, а також змінювати вміст, структуру і оформлення таких документів . При некоректній фільтрації можливо модифікувати DOM сайту, який атаковано і домогтися виконання JS-коду в контексті сайту,який атаковано.

Даний вид XSS-атаки використовує довірений, керований сервером скрипт, який надсилається користувачеві веб-додатка. Наприклад, JS-код

здійснює перевірку працездатності форми перед відправкою сервера заповнених даних користувачем. Скрипт обробляє введені дані, потім вставляє їх назад в веб-сторінку (наприклад, з Dynamic HTML). Це дає можливість реалізації «DOM XSS» тобто вбудувати шкідливий код в сценарій JS-коду.

Функція testSslCertificate перевіряє SSL сертифікат . Алгоритм цієї функції доволі простий. За допомогою додатку curl завантажується свій сертефікат безпеки та перевіряється наявність його на сайті за допомогою функцій CURLOPT_SSL_VERIFYPEER та CURLOPT_SSL_VERIFYHOST розширення curl.

Функція testDirectoryListingEnabled перевіряє на присутність відкритих для перегляду дерикторій. Працює вона наступним чином : у всіх отриманих раніше посилань відкидається частина з запитами та ставиться коса риска, після чого додаток переходить по кожному з посилань та перевіряє наявність таких ключових слов як Parent Directory або Directory. Якщо такі слова присутні - то це означає, що директорія відкрита для перегляду.

Функція testForSQLi перевіряє сайт на присутність SQL ін'єкцій. Алгоритм роботи такий : в кожне з отриманих посилань додаються одинарні або подвійні лапки, після цього, якщо на сайті присутня вразливість типу sql inj, на сторінці з'явится помилка. Саме цю помилку і відстежує додаток. Після цього отримані результати зберігаються в базі даних.

3.6 Функціонування сторінок додатку

Сайт складається с декількох головних сторінок,а саме

1) index.php - головна сторінка

2) history.php - сторінка з історією сканування (додаток д)

3) vul.php - сторінка з вразливостями та їнформацією про них (додаток е)

4) report.php - сторінка з результатами обраного сканування (додаток є)

3.6.1 Сторінка index.php

Код сторінки наданий в додатку А.

Головна сторінка сайту не містить в собі php коду, а тільки html розмітку та js скрипти, а саме beginScan() , який і викликає сторінку beginScan.php, що в свою чергу починає сканування.

```
      function beginScan(){
            $(".progress").show();
            $("#status").show();
            $("#scanstatus").show();
            var url=$("#url").val();
            $.post("beginScan.php", {link:url}, function(data){
            var id = data;
            var stop = false;
            $.post("tests.php", {testId:id}, function(data){stop = true;});
                  if (stop)$(".progress").hide();
                  var refreshId = setInterval(function() {
                  if (!stop){
      $.post("getStatus.php", {testId:id}, function(data){$("#status").html(data)});
      $.post("getPercent.php", {testId:id}, function(data){status(data)});
      $.post("getVulnerabilities.php",                                    {testId:id},
function(data){$("#scanstatus").html(data)});
                        }
            }, 1000);
      });
```

3.7 Інтерфейс сайта

Інтерфейс сайта (рис. 3.2) виконаний в мінімалістичному стилі та доволі простий і зрозумілий. На сайті присутня головна сторінка з якої можна

просканувати сайт и побачити отримані результати, сторінка з історією сканувань, сторінка з більш детальним описом вразливостей сайта, якщо такі є, та сторінка с описом вразливостей та методами їх усунення.

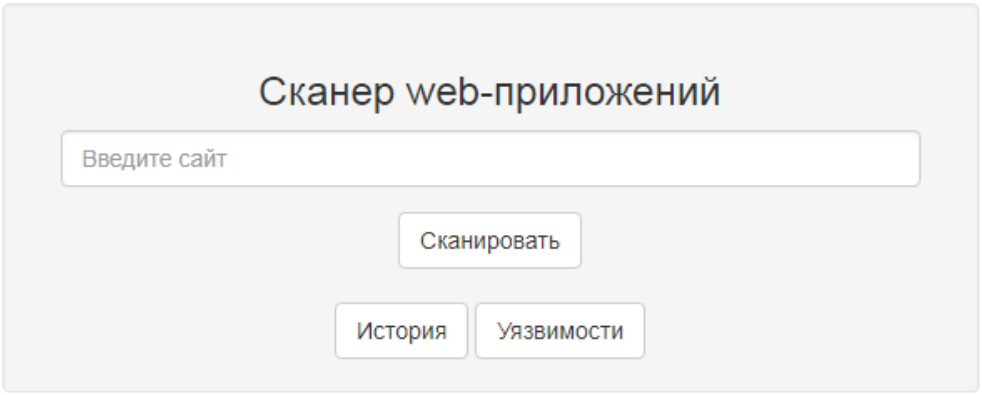


Рисунок 3.2 - інтерфейс головної сторінки до сканування

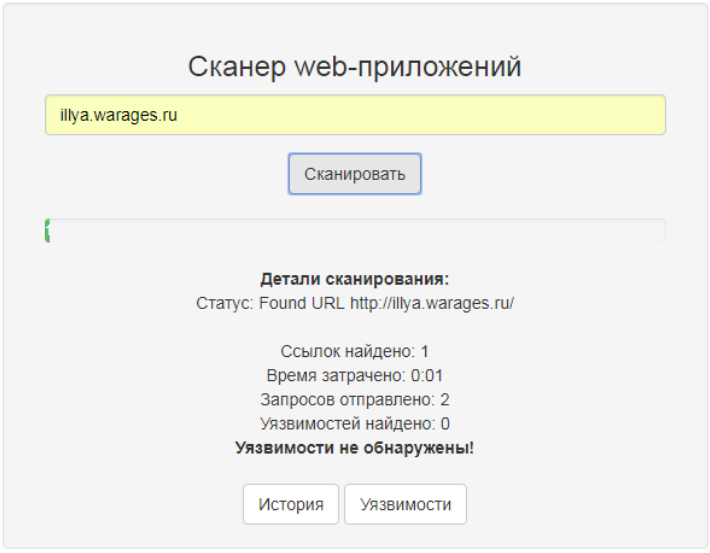


Рисунок 3.3 - інтерфейс головної сторінки під час сканування

Після сканування можно також зайти в історію сканувань та детальніше оглянути отримані результати (рис 3.4).

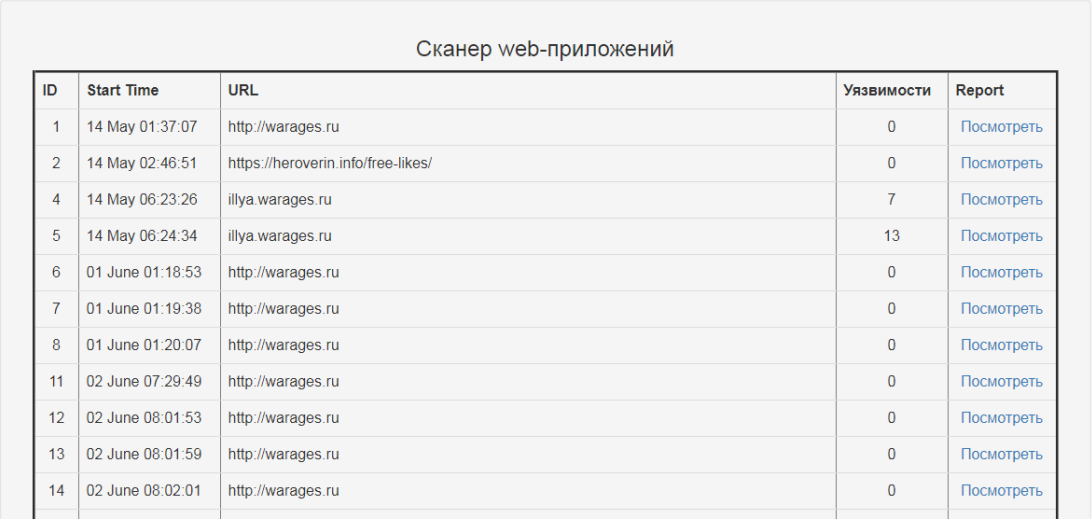


Рисунок 3.4 - інтерфейс сторінки "Історія"

Також на сайті присутня сторінка із вразливостями та рішенням, як можна усунути їх (рис 3.5).

Рисунок 3.5 - інтерфейс сторінки "Вразливості"

3.8 Інструкція корстувача

Для початку роботи з додатком потрібно обрати web - сервіс, який потрібно просканувати. Після чого на головній сторінці в поле "Введите сайт" потрібно ввести адресу сайту та натиснути кнопку "Сканировать" (рис. 3.3).

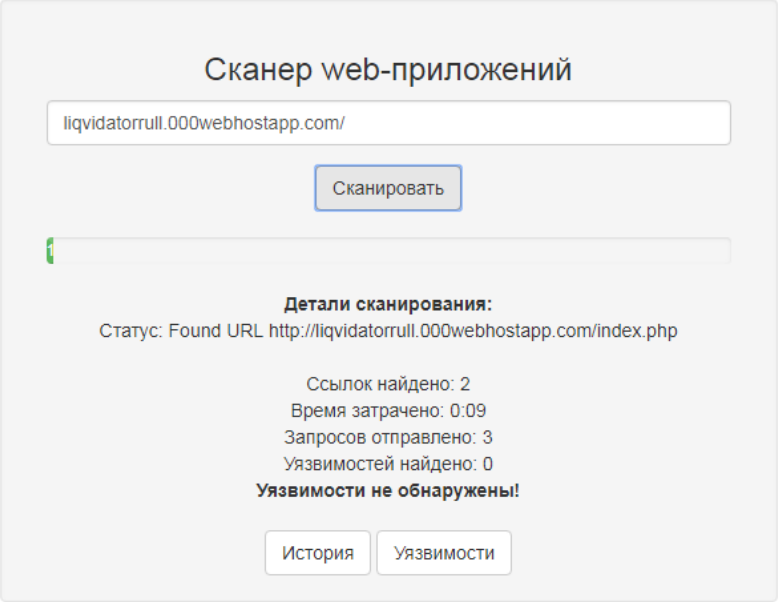


Рисунок 3.3 - початок сканування web - сервісу

Після цього почнеться сканування, яке може зайняти як 1 хвилину, так і декілька годин в залежності від складності сайту який сканується.

На сайті присутня смуга статусу, яка показує на якому етапі проходить сканування, і скільки ще потрібно очікувати.

3.9 Тестування отриманого додатку

Для тестування було розроблено та створено web сторінку із вразливостями.

Рисунок 3.1 - вигляд сайту для теста додатку

Після чого був запущений додаток для сканування (рис. 3.2)

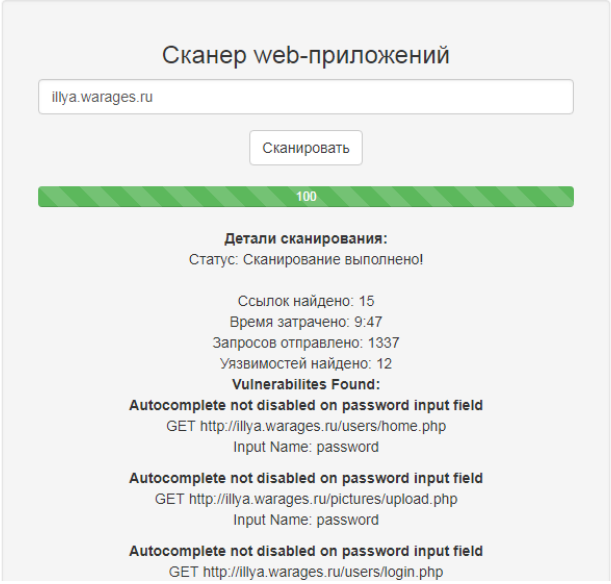


Рисунок 3.2 - успішне сканування веб - сайту для тесту

По закінченню сканування було знайдено 12 із 15 вразливостей на сайті. Також за час сканування було знайдено 15 посилань та відправлено 1337 запитів до сайту, який сканувався. На це все було затрачено 9 хвилин 47 секунд. З вразливостей було знайдено :

1) sql ін'єкція

2) автодоповнення

3) відкриті для перегляду директорії

Висновок

В даному розділі було розроблено та проаналізовано додаток для сканування web - додатків . Інструкцію по роботі з ним.

Також було розглянуто основні функції серверу та показано інтерфейс сайту .

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів, причин пожеж. Розглянуті заходи, які дозволяють забезпечити гігієну праці і виробничу санітарію. На підставі аналізу розроблені заходи з техніки безпеки та рекомендації з пожежної профілактики.

Завданням даного проекту бакалавра були інструментальні засоби оцінки і вибору засобів забезпечення надійності Web-сервісів.

4.1 Аналіз стану умов праці

Робота над створенням web - додатку для аналізу та оцінки надійності web - сервісів проходить в побутовому приміщенні. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

4.1.1 Вимоги до приміщення

Для зручності спільної роботи з іншими працівниками (обговорення ідей, з’ясування проблем і т.д.) в кімнаті є диван і журнальний стіл. Задля дотримання визначеного рівня мікроклімату в будівлі встановлено систему опалення та кондиціонування.

Згідно до санітарних норм мікроклімату виробничих приміщень [3] розмір площі для одного робочого місця оператора персонального комп’ютера має бути не менше 6 кв. м, а об’єм – неменше 20 куб. м. Отже,

дане приміщення цілком відповідає зазначеним нормам.

4.1.2 Вимоги до організації робочого місця

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця [4] і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

За ступенем пожежної безпеки приміщення належить до категорії В. Кабінет оснащений переносним вуглекислотним вогнегасником ВВК-5 .

Наявна аптечка для надання долікарської допомоги, а також у кабінеті роблять вологе прибирання та щоденно провітрюють приміщення.

4.1.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав’язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни. За ступенем нервово-психічної напруги виконання роботи можна віднести до II – III ступеня і кваліфікувати як помірно напружений – напружений за умови успішного виконання поставлених завдань.

Рекомендовано застосування екранних фільтрів, локальних світлофільтрів (засобів індивідуального захисту очей) та інших засобів захисту, а також інші профілактичні заходи [4].

4.2 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

4.2.1 Аналіз небезпечних та шкідливих факторів при розробці виробу

Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання , які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Переважно роботи за проектами виконують у кабінетах чи інших приміщеннях, де використовують різноманітне електрообладнання, зокрема персональні комп’ютери (ПК) та периферійні пристрої.

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 [4].

4.2.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання. Небезпека загоряння пов'язана з особливістю комп'ютерів – іззначною кількістю щільно розташованих на монтажній платі і блоках електронних вузлів і схем, електричних і комутаційних кабелів, резисторів,

конденсаторів, напівпровідникових діодів і транзисторів. Надійна робота окремих елементів і мікросхем в цілому забезпечується тільки в певних інтервалах температури, вологості і при заданих електричних параметрах. При відхиленні реальних умов експлуатації від розрахункових можуть виникнути пожежонебезпечні ситуації.

4.2.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

4.3 Гігієнічні вимоги до параметрів виробничого середовища

4.3.1 Мікроклімат

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [3].

Рівні позитивних і негативних іонів у повітрі мають відповідати [3]. Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

4.3.2 Освітлення

Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IVв розряду точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) рівна 200 лк. Джерелом природного освітлення є сонячне світло.

Розрахунок освітлення

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \times S_n, \qquad (4.1)$$

$$S_n = a \cdot b = 6 \cdot 4 = 24 \text{ м}^2,$$
$$S = 1/10 \cdot 24 = 2{,}4 \text{ м}^2.$$

Приймаємо 1 вікно площею S=2,4 м^2.

Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 5 м, світильниками ЛПО2П, оснащеними лампами типа

ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок кількості світильників n виробляється по формулі (4.2):

$$n=\frac{E\times S\times Z\times K}{F\times U\times M}, \qquad (4.2)$$

Підставивши числові значення у формулу (4.2), отримуємо:

$$n=\frac{300\times 24\times 1.1\times 1.5}{5400\times 0.575\times 2}=1.9$$

Приймаємо освітлювальну установку, яка складається з 3-х світильників, які складаються з 2-х люмінесцентних ламп загальною потужністю 80 Вт, напругою – 220 В.

4.3.3 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в [10].

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

4.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації

обладнання наводимо приклади деяких заходів безпеки.

1) Заходи безпеки під час експлуатації персонального комп’ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;

- експлуатацію сертифікованого обладнання;

- дотримання заходів електробезпеки;

- забезпечення оптимальних параметрів мікроклімату;

- забезпечення раціонального освітлення місця праці (освітленість робочого місця не перевищувала 2/3 нормальної освітленості приміщення);

2) Заходи безпеки під час експлуатації інших електричних приладів передбачають дотримання таких правил:

- постійно стежити за справним станом електромережі, розподільних щитків, вимикачів, штепсельних розеток, лампових патронів, а також мережевих кабелів живлення, за допомогою яких електроприлади під’єднують до електромережі;

- постійно стежити за справністю ізоляції електромережі та мережевих кабелів, не допускаючи їхньої експлуатації з пошкодженою ізоляцією;

- не тягнути за мережевий кабель, щоб витягти вилку з розетки;

- не закривати меблями, різноманітним інвентарем вимикачі, штепсельні розетки;

- не залишати включені електроприлади без нагляду;

- не ставити на електроприлади матеріали, які можуть під дією теплоти, що виділяється, загорітися (канцелярські товари, сувенірну продукцію тощо).

4.5.1 Розрахунок захисного заземлення

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [2], приміщення в якому проводяться всі роботи

відноситься до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, та 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \qquad (4.3)$$

де $R_{пр.з.}$–опір природних заземлювачів; $R_д$–допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.} = R_д$.

Підставивши числові значення у формулу (4.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40-4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ,Ом·м. Приблизне значення питомого опору глини приймаємо ρ = 40 Ом·м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр.} = \Psi \cdot \rho \qquad (4.4)$$

$$\rho_{розр.в} = 1{,}7 \cdot 40 = 68 \text{ Ом} \cdot \text{м}$$

$$\rho_{розр.в} = 5{,}5 \cdot 40 = 220 \text{ Ом} \cdot \text{м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (4.5).

$$R_В = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot 1_В} \cdot \left(\ln \frac{2 \cdot 1_В}{d_{СТ}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + 1_В}{4 \cdot t - 1_В} \right), \quad (4.5)$$

$$t = h_E + \frac{1_E}{2}, \quad (4.6)$$

$$t = 0,8 + \frac{3}{2} = 2,3 \text{ м}$$

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5$$

Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2R_E}{R_Д} = \frac{2 \times 18,5}{4} = 9,25 \quad (4.7)$$

І визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки$\eta_в$ = 0,57 (табличне значення).

Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання$\eta_в$,шт:

$$n = \frac{2 \cdot R_E}{R_Д \cdot \eta_E} = \frac{2 \cdot 18,5}{4 \cdot 0,57} \approx 16 \quad (4.8)$$

Визначається довжина з'єднувальної стрічки горизонтального

заземлювача $l_с$, м:

$$1_С=1,05\cdot L_В\cdot(n_В-1), \qquad (4.9)$$

де $L_в$– відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$– необхідна кількість вертикальних заземлювачів.

$$l_с = 1,05\cdot 3\cdot(16-1)\approx 48 \text{ м}.$$

Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки)$R_г$, Ом:

$$R_г=\frac{\rho_{розр.г}}{2\cdot\pi\cdot 1_с}\cdot\ln\frac{2\cdot 1_с^2}{d_{см}\cdot h_г}, \qquad (4.10)$$

$$R_г = \frac{220}{2\cdot\pi\cdot 48}\cdot\ln\frac{2\cdot 48^2}{0,95\cdot 0,15\cdot 0,5}=8,1 \text{ Ом}$$

Визначається коефіцієнт використання горизонтального заземлювача $\eta_с$. відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги $\eta_с$ = 0,3.

Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг.}=\frac{R_E\cdot R_г}{R_E\cdot\eta_с+R_г\cdot n_E\cdot\eta_E}\le R_д, \qquad (4.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг} < 4$ Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \leq R_{д}$$

Висновки до розділу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ПЕРЕЛІК ПОСИЛАНЬ ДО РОЗДІЛУ 4

НПАОП 0.00.-1.28-10 «Правил охорони праці під час експлуатації електронно-обчислювальних машин»;

НПАОП 0.00-4.15-98 «Положення про розробку інструкцій з охорони праці»;

ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» ;

ДСанПіН 3.3.2.007-98 «Правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»;

ГОСТ 12.1.044-89 «ССБТ. Вогнестійкість. Номенклатура показників і методи їх визначення»;

ГОСТ 12.1.030-81 «Електробезпека. Захисне заземлення, занулення».

ГОСТ 13109-97«Електрична енергія. Сумісність технічних засобів. Норми якості електричної енергії в системах електропостачання загального призначення»;

ДБН В.2.5-28:2015 «Державні Будівельні Норми України. Природне і штучне освітлення»;

НАПБ Б.03.002-2007 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою».

ДБН В.2.5-67:2013. «Опалення, вентиляція та кондиціонування.»

Нечаев И. Индикатор металлических предметов. // Радио, 2003, №10, с. 56.

ВИСНОВОК

В даній роботі були розглянуті основні вразливості Web-сайтів та серверів і наведено їх приклади. Також було розглянуто ієрархію та рівні захисту Web - серверів.

Далі були розглянуті методи пошуку вразливостей. Була підведена статистика вразливостей та їх поширеність в різних сферах.

Наступним кроком були розглянуті та проаналізовані способи аналізу захищеності web - додатків.

Після чого було сформовано вимоги до розроблюваного web - додатка.

3 розділ присвячений розробці власного сайту для аналізу web - додатків на вразливості, а саме :

1) Обгрунтування вибору технологій

2) Формуваннія вимог до системи

3) Реалізація серверу

4) Розробка серверу

5) Інтерфейс сайту

Таким чином було розроблено та протестовано web - додаток для сканування та аналізу вразливостей для подальшого їх усунення.

СПИСОК ЛІТЕРАТУРИ

1. "A new spoof: all frames-based sites are vulnerable" - SecureXpert Labs http://tbtf.com/archive/11-17-98.html#s02
2. Cross Site Scripting Info http://httpd.apache.org/info/css-security/
3. “XSS без XSS”. By Marsel [marsel roses.ru] aka Phoenix http://www.securitylab.ru/contest/212115.php
4. HTTP REQUEST SMUGGLING, http://www.securitylab.ru/analytics/216403.php
5. "CRLF Injection" by Ulf Harnhammar (BugTraq posting), http://www.securityfocus.com/archive/1/271515
6. Brute Force Attack", Imperva Glossary http://www.imperva.com/application_defense_center/glossary/brute_force.html
7. "iDefense: Brute-Force Exploitation of Web Application Session ID's", By David Endler - iDEFENSE Labs http://www.cgisecurity.com/lib/SessionIDs.pdf
8. "Protecting Secret Keys with Personal Entropy", By Carl Ellison, C. Hall, R. Milbert, and B. Schneier http://www.schneier.com/paper-personal-entropy.html
9. "Emergency Key Recovery without Third Parties", Carl Ellison http://theworld.com/~cme/html/rump96.html
10. "Best Practices in Managing HTTP-Based Client Sessions", Gunter Ollmann - X-Force Security Assessment Services EMEA http://www.technicalinfo.net/papers/WebBasedSessionManagement.html
11. "A Guide to Web Authentication Alternatives", Jan Wolter http://www.unixpapa.com/auth/homebuilt.html
12. "Dos and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Computer Science

http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf

13. "Session Fixation Vulnerability in Web-based Applications", By Mitja Kolsek - Acros Security
    http://www.acrossecurity.com/papers/session_fixation.pdf
14. "Divide and Conquer", By Amit Klein – Sanctum
    http://www.sanctuminc.com/pdf/whitepaper_httpresponse.pdf
15. "Smashing The Stack For Fun And Profit", By Aleph One - Phrack 49
    http://www.insecure.org/stf/smashstack.txt
16. «ОШИБКИ ПЕРЕПОЛНЕНИЯ БУФЕРА ИЗВНЕ И ИЗНУТРИ КАК ОБОБЩЕННЫЙ ОПЫТ», Крис Касперски
    http://www.samag.ru/art/03.2004/03.2004_07.pdf
17. "Understanding LDAP"
    http://www.redbooks.ibm.com/redbooks/SG244986.html
18. «Внедрение SQL кода с завязанными глазами», Офер Маор, Амичай Шалман http://www.securitylab.ru/analytics/216332.php
19. PhpInclude.Worm
    http://www.frsirt.com/exploits/20041225.PhpIncludeWorm.php
20. "Blind XPath Injection" - Amit Klein
    http://www.sanctuminc.com/pdfc/WhitePaper_Blind_XPath_Injection_20040518.pdf
21. Directory Indexing Vulnerability Alerts
    http://www.securityfocus.com/bid/1063
22. "Hypertext Transfer Protocol -- HTTP/1.1" http://www.cis.ohio-state.edu/cgi-bin/rfc/rfc2068.html#sec-14.39
23. "Novell Groupwise Arbitrary File Retrieval Vulnerability"
    http://www.securityfocus.com/bid/3436/info/
24. "FormMail Real Name/Email Address CGI Variable Spamming Vulnerability" http://www.securityfocus.com/bid/3955
25. "Ravaged by Robots!", By Randal L. Schwartz

http://www.webtechniques.com/archives/2001/12/perl/

26."Dos and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Computer Science http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf

ДОДАТОК А

Головна сторінка сайту

```
     <link rel="stylesheet" href=" bootstrap.min.css" />
     <script type="text/javascript" src=" jquery-1.11.3.min.js"></script>
     <script type="text/javascript" src=" bootstrap.min.js"></script>
      <div class="container">
      <div class="row">
      <div class="col-md-6 col-md-offset-3 well">
      <h3 class="text-center">Сканер web-приложений</h3>
      <form class="form">
      <div class="col-xs-12">
      <div class="form-group">
      <input      type="text"      class="form-control"      name="url"      id="url"
placeholder="Введите сайт" />
      </div>
      </div>
      <div class="text-center col-xs-12">
      <input     type="button"     class="btn     btn-default"     value="Сканировать"
onclick="beginScan()" /><br><br>
         <div class="progress" id="barbox" style="display:none;">
             <div  id="bar"  class="progress-bar  progress-bar-success  progress-bar-
striped" role="progressbar"  aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">
             <span id="count">0</span>
             </div>
         </div>
      <div id="status" style="display:none;"></div>
      <div id="scanstatus" style="display:none;"></div>
      <a  href=""  class="btn  btn-default">История</a>  <a  href=""  class="btn  btn-
default">Уязвимости</a>
      </div>
      </form>
      </div>
      </div>
      </div>
      </div>
      <script>
      function beginScan(){
            $(".progress").show();
            $("#status").show();
            $("#scanstatus").show();
```

```
            var url=$("#url").val();
            $.post("beginScan.php", {link:url}, function(data){
            var id = data, stop = false;
            $.post("tests.php", {testId:id}, function(data){stop = true;});
                  if (stop)$(".progress").hide();
                  var refreshId = setInterval(function() {
                  if (!stop){
      $.post("getStatus.php", {testId:id}, function(data){$("#status").html(data)});
      $.post("getPercent.php", {testId:id}, function(data){status(data)});
      $.post("getVulnerabilities.php",                                    {testId:id},
function(data){$("#scanstatus").html(data)});
                        }}, 1000);
      });
            function status(percent){
            var box=percent;
            var main = box * 100;
            var value= (main / 100);
            var count= box;
            if(box <= 100)
            {
            $('#count').html(count);
            $('#bar').animate(
            {
            "width": value+'%',
            }, 1);
            }else{
     $('#bar').attr({ class: 'progress-bar progress-bar-danger progress-bar-striped'
});
            }
            return false;

      }

      $.ajaxSetup({ cache: false });

     }
     </script>
```

ДОДАТОК Б

begin_scan.php

```
<?
set_time_limit(0);
error_reporting(0);

$currentDir = './';
require_once($currentDir . 'databaseFunctions.php');
$connectionFlag = connectToDb($db);
$urlToScan = $_POST['link'];

if(empty($urlToScan))
{
 echo 'Вы не ввели адрес сайта!';
 return;
}

if(stripos($urlToScan, 'http') !== 0)
 $urlToScan = 'http://' . $urlToScan;

$nextId = generateNextTestId($db);

                if(!$nextId)
                {
                        echo 'Ошибка генерации NEXTID';
                        return;
                }
                else
                {
                        $testId = $nextId;
                        $now = time();
                        $query              =              "INSERT              into
tests(id,status,numUrlsFound,type,num_requests_sent,start_timestamp,finish_timestamp,
scan_finished,url,username,urls_found,percent)  VALUES($nextId,'Creating  profile  for
new scan...',0,'scan',0,$now,$now,0,'$urlToScan','','',0)";

                        $result = $db->query($query);
                        if(!$result)
                        {
                        echo 'Проблема в выполнении запроса. Попробуйте еще раз';
```

```
                                return;
                        }
                }

                updateStatus($db, 'Запуск...', $testId);
                updatePercent($db, 1, $testId);
                $query = "UPDATE tests SET numUrlsFound = 0 WHERE id = $testId;";
                $db->query($query);
                $query = "UPDATE tests SET duration = 0 WHERE id = $testId;";
                $db->query($query);

    echo $testId;

    ?>
```

ДОДАТОК В

/tests/testSSLCerteficate.php

```
    <?php
    set_time_limit(0);

    function testSslCertificate($urlsToTest, $testId){

    connectToDb($db);
    updateStatus($db, "Testing $urlsToTest for untrustworthy SSL certificates...",
$testId);

    updateStatus($db, "Identifying which URLs, if any, begin with HTTPS...",
$testId);

    $usingHttps = false;
    $httpsUrl = '';

    foreach($urlsToTest as $currentUrl)
    {
     if(substr($currentUrl, 0, 5) == 'https')
     {
            $usingHttps = true;
            $httpsUrl = $currentUrl;
            echo "https url = $currentUrl <br>";
            break;
```

```
 }
}

if($usingHttps)
{
 $http = new http_class;
 $http->timeout=0;
 $http->data_timeout=0;
 $http->user_agent="Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)";
 $http->follow_redirect=1;
 $http->redirection_limit=5;

 $cacertsUrl = "http://curl.haxx.se/ca/cacert.pem";

 $error=$http->GetRequestArguments($cacertsUrl,$arguments);

 $error=$http->Open($arguments);

 if($error=="")
 {
        $error=$http->SendRequest($arguments);

        if($error=="")
        {
               $headers=array();
               $error=$http->ReadReplyHeaders($headers);
               if($error=="")
               {
                      $responseCode = $http->response_status;
                      if(intval($responseCode) == 200)
                      {
                             $cacerts = file_get_contents($cacertsUrl);
                             $oldCacerts = file_get_contents('tests/cacert.pem');
                             if($cacerts != $oldCacerts)
                             {

 file_put_contents('tests/cacert.pem',$cacerts);
                             }
                      }
               }
```

```
            }
      }

      $ch = curl_init();

      curl_setopt($ch, CURLOPT_URL, $httpsUrl);
      curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
      $user_agent = "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)";
      curl_setopt($ch, CURLOPT_USERAGENT, $user_agent);
      curl_setopt( $ch, CURLOPT_FOLLOWLOCATION, true );
      curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, true);
      curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);
      curl_setopt($ch, CURLOPT_CAINFO, getcwd() . "/cacert.pem");
      $response = curl_exec($ch);

      if($db)incrementHttpRequests($db, $testId);

      if(empty($response))
      {
      echo '<br>SSL Certificate is not trusted!<br>Url: ' . $httpsUrl . '<br>';
            echo 'Method: GET <br>';
            echo 'Error: ' .  curl_error($ch) . '<br>';
            $tableName = 'test' . $testId;
            $query = "SELECT * FROM test_results WHERE test_id = $testId AND type =
'sslcert' AND method = 'get' AND url = '$httpsUrl' AND attack_str = '$httpsUrl'";
            $result = $db->query($query);
            if($result)
            {
                  $numRows = $result->num_rows;
                  if($numRows == 0)
                  {
            insertTestResult($db, $testId, 'sslcert', 'get', $httpsUrl, $httpsUrl);
                  }
            }
      }
         curl_close($ch);
     }

     }
```

ДОДАТОК Г

/classes/databaseFunctions.php

```
<?php

function connectToDb(&$db)
{
 $db = $db = new mysqli( 'localhost', 'root', '', 'webvulscan');
 if (mysqli_connect_errno()) {
       return false;
 }
 return true;
}

function updateStatus($db, $newStatus, $testId)
{
 $query = "UPDATE tests SET status = '$newStatus' WHERE id = $testId;";
 $result = $db->query($query);
 return $result;
}

function updatePercent($db, $newPercent, $testId)
{
 $query = "UPDATE tests SET percent = '$newPercent' WHERE id = $testId;";
 $result = $db->query($query);
 return $result;
}

function insertTestResult($db, $testId, $type, $method, $url, $attackStr)
{
 $query = "INSERT into test_results(test_id, type, method, url, attack_str)
VALUES($testId,'$type','$method','$url','$attackStr')";
 $result = $db->query($query);
 return $result;
}

function generateNextTestId($db)
{
 $query = "SELECT MAX(id) FROM tests";
 $result = $db->query($query);
```

```
      if(!$result)
            return $result;

      $row = $result->fetch_array();

      $maxId = $row[0] + 1;
      //$maxId = $row->id;//or else $row->MAX(id)
      return $maxId;
     }

     function incrementHttpRequests($db, $testId)
     {
      $query = "UPDATE tests SET num_requests_sent = (num_requests_sent + 1) WHERE
id = $testId";
      $result = $db->query($query);
      return $result;
     }



     function getSiteBeingTested($urlToScan)
     {
      $countSlashes = 0;
      $lastIndexOfSlash = 0;
      $lastIndexOfDot = 0;
      $siteBeingTested = $urlToScan;
      for($i=0; $i<strlen($siteBeingTested); $i++)
      {
            if($siteBeingTested[$i] == '/')
            {
                  $lastIndexOfSlash = $i;
                  $countSlashes++;
            }
            if($siteBeingTested[$i] == '.')
            {
                  $lastIndexOfDot = $i;
            }
      }
      if(($lastIndexOfDot < $lastIndexOfSlash) && (substr($siteBeingTested, -1) !=
'/'))
            $siteBeingTested .= '/';
```

```
      else if($countSlashes > 2)
            $siteBeingTested = substr($siteBeingTested, 0 , $lastIndexOfSlash+1);

      return $siteBeingTested;
     }



     ?>
```

ДОДАТОК Д

vul.php

```
     <?php

     $currentDir = './';
     require_once($currentDir . 'databaseFunctions.php');
     connectToDb($db);
     ?>

     <link                                                    rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
      <script                                               type="text/javascript"
src="http://bootstraptema.ru/plugins/jquery/jquery-1.11.3.min.js"></script>
     <script  type="text/javascript"  src="http://bootstraptema.ru/plugins/2015/b-v3-
3-6/bootstrap.min.js"></script>

      <br> <br> <br> <br> <br><div class="container">
      <div class="row">
      <div class="col-md-12 well">
      <h3 class="text-center">Сканер web-приложений</h3>
      <form class="form">
      <div class="col-xs-12">

     <?
     $query = "SELECT * FROM vulnerabilities ";
      //echo $query;
      $result = $db->query($query);
      if($result)
      {
            $numRows = $result->num_rows;
```

```
                    echo                        '<table                        class="table"
border="3"><tr><th>Name</th><th>Description</th><th>Solution</th><th>Danger</th></tr>
';
                    for($i=0; $i<$numRows; $i++)
                    {
                           $row = $result->fetch_object();
                           $name = $row->name;
                           $description = $row->description;
                           $solution = $row->solution;
                           $danger = $row->priority;

                           $numVulns = 'Unknown';
                           $query = "SELECT * FROM test_results WHERE test_id = $id";
                           $resultTwo = $db->query($query);
                           if($resultTwo)
                                  $numVulns = $resultTwo->num_rows;

                           $report   =   '<a   href="report.php?id='   .   $id   .   '"
target="_blank">Посмотреть</a>';

                           echo '<tr>';
                           echo "<td align='center'>$name</td>";
                           echo "<td align='left'>$description</td>";
                           echo "<td align='left'>$solution</td>";
                           echo "<td align='center'>$danger</td>";
                           echo '</tr>';

                    }
                    echo '</table>';

      }

            ?>

      <a  href="history.php"  class="btn  btn-default">История</a>  <a  href="vul.php"
class="btn btn-default">Уязвимости</a>

      </div>
      </form>
      </div> </div> </div> </div>
```

ДОДАТОК Е

history.php

```
<?php

$currentDir = './';
require_once($currentDir . 'databaseFunctions.php');
connectToDb($db);
?>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
 <script type="text/javascript" src="http://bootstraptema.ru/plugins/jquery/jquery-1.11.3.min.js"></script>
<script type="text/javascript" src="http://bootstraptema.ru/plugins/2015/b-v3-3-6/bootstrap.min.js"></script>

 <br> <br> <br> <br><div class="container">
 <div class="row">
 <div class="col-md-12 well">
 <h3 class="text-center">Сканер web-приложений</h3>
 <form class="form">
 <div class="col-xs-12">

<?
$query = "SELECT * FROM tests WHERE type = 'scan'";
 //echo $query;
 $result = $db->query($query);
 if($result)
 {
	$numRows = $result->num_rows;
	if($numRows == 0)
		echo 'Сканирований нет!';
	else
	{
		echo '<table class="table" border="3"><tr><th>ID</th><th>Start Time</th><th>URL</th><th>Уязвимости</th><th>Report</th></tr>';
		for($i=0; $i<$numRows; $i++)
		{
			$row = $result->fetch_object();
			$id = $row->id;
```

```
                        $startTime = $row->start_timestamp;
                        $startTimeFormatted = date('d F h:i:s', $startTime);
                        $url = $row->url;

                        $numVulns = 'Unknown';
                        $query = "SELECT * FROM test_results WHERE test_id = $id";
                        $resultTwo = $db->query($query);
                        if($resultTwo)
                              $numVulns = $resultTwo->num_rows;

                        $report = '<a href="report.php?id=' . $id . '"
target="_blank">Посмотреть</a>';

                        echo '<tr>';
                        echo "<td align='center'>$id</td>";
                        echo "<td align='left'>$startTimeFormatted</td>";
                        echo "<td align='left'>$url</td>";
                        echo "<td align='center'>$numVulns</td>";
                        echo "<td align='center'>$report</td>";
                        echo '</tr>';

                  }
                  echo '</table>';

            }

      }

            ?>

      <a href="history.php" class="btn btn-default">История</a> <a href="vul.php"
class="btn btn-default">Уязвимости</a>

      </div>
      </form>
      </div>
      </div>
      </div>
      </div>
```

ДОДАТОК Є

report.php

```
<?php

$currentDir = './';
require_once($currentDir . 'databaseFunctions.php');
connectToDb($db);
?>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" />
 <script type="text/javascript" src="http://bootstraptema.ru/plugins/jquery/jquery-1.11.3.min.js"></script>
<script type="text/javascript" src="http://bootstraptema.ru/plugins/2015/b-v3-3-6/bootstrap.min.js"></script>

 <br> <br> <br> <br><div class="container">
 <div class="row">
 <div class="col-md-6 col-md-offset-3 well">
 <h3 class="text-center">Сканер web-приложений</h3>
 <form class="form">
 <div class="col-xs-12">

<?
isset($_GET['id']) ? $testId = $_GET['id'] : $testId = 0;

$query = 'SELECT * FROM test_results WHERE test_id = ' . $testId;
connectToDb($db);
$result = $db->query($query);
if($result)
{
 $numRows = $result->num_rows;
 if($numRows > 0)
 {
        echo '<b>Уязвимости найдены : </b><br><br>';

        for($i=0; $i<$numRows; $i++)
        {
               $row = $result->fetch_object();
```

```
$type = $row->type;
$method = strtoupper($row->method);
$url = $row->url;
$info = $row->attack_str;

if($type == 'rxss')
{
	$type = 'Reflected Cross-Site Scripting';
	$info = 'Query Used: ' . $info;
}
else if($type == 'sxss')
{
	$type = 'Stored Cross-Site Scripting';
	$info = 'Query Used: ' . $info;
}
else if($type == 'sqli')
{
	$type = 'SQL Injection';
	$info = 'Query Used: ' . $info;
}
else if($type == 'idor')
{
	$type = '(Potentially Insecure) Direct Object Reference';
	$info = 'Object Referenced: ' . $info;
}
else if($type == 'basqli')
{
	$type = 'Broken Authentication using SQL Injection';
	$info = 'Query Used: ' . $info;
}
else if($type == 'unredir')
{
	$type = 'Unvalidated Redirects';
	$info = 'URL Requested: ' . $info;
}
else if($type == 'dirlist')
{
	$type = 'Directory Listing enabled';
	$info = 'URL Requested: ' . $info;
```

```
                    }
                    else if($type == 'bannerdis')
                    {
                          $type = 'HTTP Banner Disclosure';
                          $info = 'Information Exposed: ' . $info;
                    }
                    else if($type == 'autoc')
                    {
                    $type = 'Autocomplete not disabled on password input field';
                          $info = 'Input Name: ' . $info;
                    }
                    else if($type == 'sslcert')
                    {
                          $type = 'SSL certificate is not trusted';
                          $info = 'URL Requested: ' . $info;
                    }

                    echo "<p><b>$type</b><br>";
                    $urlHtml = htmlspecialchars($url);
                    echo "$method $urlHtml<br>";
                    $infoHtml = htmlspecialchars($info);
                    echo "$infoHtml</p>";
             }
             $result->free();
             $db->close();
      }
      else
      {
             echo '<b>Уязвимости не обнаружены!</b><br><br>';
      }
     }
             ?>
      <a  href="history.php"  class="btn  btn-default">История</a>  <a  href="vul.php"
class="btn btn-default">Уязвимости</a>
      </div>
      </form>
      </div>
      </div>
      </div>
      </div>
```

ДОДАТОК Ж

Комп’ютерна презентація

Інструментальні засоби оцінки і вибору засобів забезпечення надійності Web-сервісів

Керівник дипломної роботи: доц. Недзельский Дмитро Олександрович
Студент: Покришка Сергій Анатолійович

Актуальність теми і мета проекту

- Стрімкий ріст IT індустрії призводить до появи нових вразливостей.
- Автоматизація пошуку вразливостей та їх аналіз значно полегшує роботу з забезпеченням захисту web – додатків
- Забезпечення надійності web – додатку значно зменшує фінансові від взломів

Постановка задачі

- Вивчення і аналіз вразливостей web – додатків.
- Практичний аналіз вразливостей та їх усунення.
- Аналіз статистики вразливостей в різних сферах.
- Розробка додатку для сканування та аналізу web – додатків .

Рішення поставленої задачі

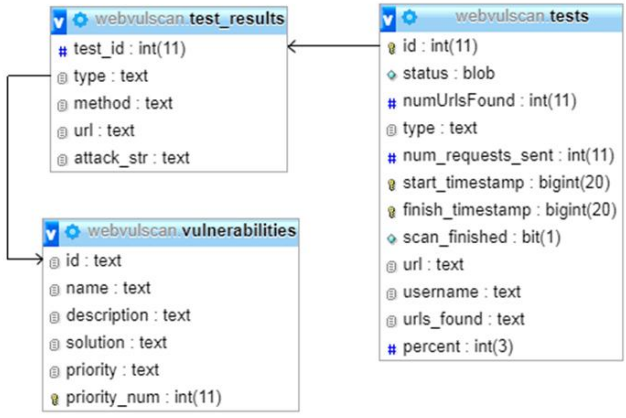
В результаті розробки був отриманий додаток для сканування та аналізу надійності web - сервісів. Розроблена система сканує обраний сайт, виводить його вразливості ,зберігає їх для подальшої обробки та надає рекомендації щодо усунення вразливостей та забезпечення надійності.

Інструменти розробки

- Сайт був розроблений за допомогою типових засобів веб-розробки.
- Основою сайту є HTML 5 – мова розмітки вмісту веб-сторінок.
- CSS 3 відповідає за зовнішній вигляд сайту.
- MySQL – реляційна СУБД, за допомогою якої була розроблена та функціонує основна частина сайту – база даних.
- PHP 7 – серверна скриптова мова програмування, за допомогою якої формуються клієнтські веб-сторінки.

База даних

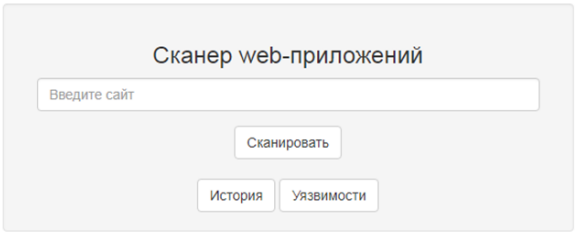
Для функціонування сайту був розроблений комплекс зв'язаних таблиць даних, які управляються за допомогою СУБД MySQL



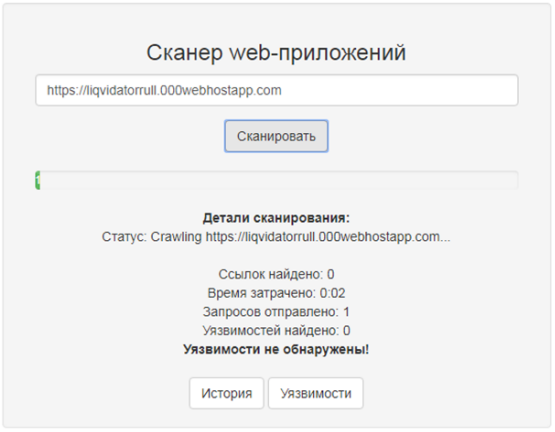
Реалізація

- Сайт буде знаходитись у відкритому доступі у Інтернеті і буде доступний як зі стаціонарних, так і з мобільних пристроїв.
- Всі результати тестів зберігаються на сайті для подальшої обробки чи аналізу
- Можна обрати лише деякі вразливості, на наявність яких потрібно просканувати сайт

Скріншоти сторінок сай



- Головна сторінка сайту



- Безпосередньо саме сканування

Результати роботи

- В результаті розробки було реалізовано додаток для сканування web – сервісів . Дана система надає можливість просканувати сайт та отримати детальний аналіз вразливостей, які були знайдені.
- Дана система була побудована за допомогою найпопулярніших на сьогодні засобів розробки: HTML, CSS, JavaScript, PHP 7 та СУБД MySQL 5.1.
- Всі функції сканування вразливостей поділені в різні файли, що дозволяє легко розширювати можливості сканеру.
- Максимально простий інтерфейс з яким розбереться будь-хто

Висновки

- В дипломній роботі були проаналізовані основні вразливості, був проведений їх аналіз та статистика . За допомогою інструментів веб-розробки, таких як HTML, CSS, JavaScript, PHP та СУБД MySQL реалізовано сканер вразливостей web - сервісів.
- Розроблена система дозволяє сканувати сайт на такі поширені вразливості як : XSS , SQL inj. Також перевіряє відкриті для перегляду директорії та SSL сертифікат
- Надалі, розвиваючи сканер, можна додавати інші вразливості для розширення функціоналу та залучення нової аудиторії, яка буде використовувати сканер