МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук

«_____»_____ 20__ р.

МАГІСТЕРСЬКА АТЕСТАЦІЙНА РОБОТА

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

Тема: Програмний модуль захисту форм авторизації веб-сайтів

Автор: М.В. Мірошниченко

Науковий керівник: к.т.н., доцент А.Б. Єлізаров

Нормоконтролер: к.т.н., доцент А.Б. Єлізаров

Київ 2020

ВСТУП

Актуальність. У зв’язку зі стрімким зростом переходу усіх аналогових даних у цифрову площину, інтернет з кожним днем наповнюється гігантською кількість нових даних, що можуть нести цінність. Особисті дані інтернет користувачів, які зберігаються у базах даних сайтів банків, інтернет-магазинів, соціальних мереж та інших підприємств, без належного захисту можуть опинитися у руках зловмисників і привести до значних збитків як бізнесу, так і звичайних користувачів.

Оскільки переважна більшість бізнесів для успішної діяльності вимушені мати веб-сайти, то гостро постає питання захисту даних їх користувачів. Поріг мінімальних знань для проведення хакерської атаки знижується із кожним днем, і теперь майже будь-яка заінтересована в цьому людина може використаті відкриті дані щодо методів проведення хакерських атак аби нанести шкоду. Постає необхідність підвищення захисту особистих даних користувачів, які зазвичай вносяться при реєстрації на сайтах. Зазвичай, доступ до цих даних звичайний користувач, так само як і порушник у разі успішної атаки, можуть отримувати авторизувавшись до особистого кабінету.

Для ефективного захисту необхідно розробити програмний модуль захисту форм авторизації веб-сайтів, що дозволить невеликим бізнесам, які не мають значних коштів на послуги професійних компаній з інформаційної безпеки, підвищити захищеність даних своїх користувачів. Для цього необхідно проаналізувати технології та вразливості веб-сайтів, аби розробити якісний захист, що не буде шкодити користувацькому досвіду реальних людей. На основі проведеного аналізу технологій та їх вразливостей, необхідно розробити методику проведення експериментальних досліджень для отримання повної інформації, що знадобиться для задачі розробки алгоритму захисту та його програмного коду. Розроблений програмний модуль повинен бути достатньо простим для того, щоб невеликий бізнес мав змогу їм користуватись, і достатньо ефективним для того, щоб захистити дані користувачів.

Відомі підходи до вирішення поставленої задачі. На сьогоднішній день існує багато інструментів захисту веб-сайтів. Зазвичай вони розробляються для конкретної CMS або фреймворку. Часто перед фахівцями компанії для підвищення ефективності вирішення завдань захисту особистих даних користувачів виникають питання про відповідність доступних програмних модулів до саме їх системи, чи відповідають вони поставленій заадачі і чи будуть сумісними та коректно працювати. Ці та інші фактори створюють труднощі у виборі програмного модулю захисту форм авторизації.

Метою роботи є розробка програмного модуля для забезпечення захисту форм авторизації від хакерських атак методом повного перебору.

Для досягнення поставленої мети вирішуються такі **задачі:**

– аналіз сучасних технологій роботи веб-сайтів та їх вразливостей;

– розробка методу вирішення задачі захисту форм авторизації веб-сайтів;

– розробка алгоритму та програмного модулю захисту форм авторизації на основі результатів експериментів.

Галузь застосування. Розроблений метод та програмний модуль відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності даних користувачів сучасних веб-сайтів за рахунок забезпечення захисту форм авторизації від хакерських атак.

Об’єктом дослідження є процес захисту веб-сайтів від хакерських атак.

Предметом дослідження є методи захисту форм авторизації від атак методом повного перебору.

Методи дослідження базуються на основі проведення експериментальних атак (для отримання даних, що дозволять розробити алгоритм захисту), та об’єктно-орієнтованого програмування (для програмної реалізації розробленого алгоритму).

Новизна одержаних результатів полягає в наступному:

– Розроблено програмний модуль захисту форм авторизації веб-сайтів, що не потребує значних знань та часу для його встановлення, але натомість є

ефективним від атак повного перебору і при цьому не погіршує користувацький досвід, тобто не створює незручностей для реальних користувачів.

Практичне значення отриманих результатів:

- Створено програмний модуль мовою php, що дозволяє підвищити безпеку веб-сторінок із формою авторизації від атаки методом повного перебору
- Створено програмний код мовою php, що дозволяє проводити експериментальне дослідження атак методом повного перебору та отримувати на веб-сторінці таблиці із результатами експерименту

Апробація. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- XVI Міжнародна науково-практична конференція «Наукові горизонти» (Шеффілд, 2020), тема доповіді: «Програмний модуль захисту форм авторизації веб-сайтів»
- XVI Міжнародна науково-практична конференція «Прикладні наукові розробки» (Прага, 2020), тема доповіді: «Програмні методи захисту локальної мережі на базі обладнання CISCO»

Розділ 1. ОГЛЯД СУЧАСНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ І ВИКОРИСТАННЯ ВЕБ-САЙТІВ ТА АНАЛІЗ ЇХ ВРАЗЛИВОСТЕЙ

1.1. Технології для створення та експлуатації веб-сайтів

Питання вразливостей веб-сайтів потребує фундаментального розуміння того, як ці сайти влаштовані. Проведемо огляд літератури щодо технологій, що безпосередньо використовуються сучасними сайтами. Розглянемо такі теми: HTML, CSS, JavaScript, PHP, бази даних SQL, сервери, вразливості веб-сайтів, CAPTCHA.

Для створення сайтів використовується HTML – мова розмітки тегів. Новітньою актуальною версією є HTML5. Особливості HTML5 у порівнянні з минулими версіями:

Здебільшого сумісний з тим, що там вже є - не потрібно вчити абсолютно нову мову для використання HTML5. Нові засоби розмітки працюють таким же чином, як і старі (хоча семантика деяких елементів змінилася) і нові API ґрунтуються здебільшого на тому ж JavaScript / DOM, який розробники програмували протягом багатьох років.

- Додає нові потужні засоби в HTML, які були раніше доступні в Web тільки за допомогою технології плагінів, такі як Flash, або за допомогою складного коду JavaScript і спеціальних прийомів. Перевірка форм і відео є кращими прикладами.
- Краще підходить для написання динамічних додатків, ніж попередні версії HTML (HTML був створений спочатку для створення статичних документів).
- Має чітко визначений алгоритм синтаксичного аналізу, так що всі браузери, які реалізують HTML5, будуть створювати однакове дерево DOM з

однієї і тієї ж розмітки, незалежно від правильності. Це величезний виграш для сумісності.

Нові елементи HTML5:

<header>: Використовується для заголовка сайту.

<footer>: Використовується для нижнього колонтитула сайту.

<nav>: Містить навігаційні функції сторінки.

<article>: Містить автономний фрагмент контенту, який буде мати сенс, якщо використовується як позиція RSS, наприклад, новинне повідомлення.

<section>: Використовується або для об'єднання в групу різних статей з різною метою або з різних тем, або для визначення різних розділів однієї статті.

<time>: Використовується для розмітки часу і дати.

<aside>: Визначає блок контенту, який пов'язаний з основним контентом, але не входить в його основний потік.

<hgroup>: Використовується в якості оболонки приховування більше одного заголовка, якщо потрібно, щоб враховувався тільки один заголовок в структурі заголовків сторінки.

<figure> і <figcaption>: Використовується для інкапсуляції малюнка як єдиного елемента, і містить, відповідно, підпис для малюнка[1].

В HTML усі теги являють собою ієрархічну систему. Вони можуть бути вкладені тільки в певні теги. Так, найпершим тегом в HTML-документі повинен бути HTML, оголошення якого йде відразу після DOCTYPE. У цьому тезі можуть перебувати тільки теги HEAD або BODY. Сам документ повинен розташовуватися всередині тега BODY, а інформація, що описує документ, - в тезі HEAD.

Теги - це керуючі команди, що говорять браузеру про те, як саме потрібно вивести на екран частина тексту, укладену в тезі. Фактично тег - це певна послідовність символів, перед якою йде символ < і після якої йде символ >, тобто виглядає це так: <тег>. Якщо браузер зустріне будь-який невідомий йому тег, то проігнорує його. Валідатор же повідомить про помилку, якщо в HTML-документі

будуть виявлені теги, що не входять в обрану специфікацію HTML. Теги бувають одинарні і парні.

- Одинарний тег. Складається тільки з однієї послідовності символів - <тег>, тобто в нього не укладено жодної текст. Одинарному тегу текст не потрібен. Він служить для інших цілей - переведення рядка, виведення картинок, виведення елементів форми і т. д.
- Парний тег. Парні теги ще називають контейнерами. Вони мають такий вигляд: <тег> </ тег>, тобто відкриває тег - <тег> і закриває тег - </ тег>.

Усередині одинарного або відкриваючого тега можуть перебувати атрибути і їх значення: <тег атрибут1 = "значення1" атрибут2 = "значення2"> текст </ тег>. Атрибути і їх значення необхідні для того, щоб будь-яким чином змінити функціональність того чи іншого тега. Порядок розташування атрибутів в тезі ні на що не впливає. Кілька атрибутів всередині одного тега розділяються пробілами. Атрибути можна також переносити на новий рядок:

<тег атрибут1 = "значення1"
атрибут2 = "значення2"> текст </ тег>

Що ж таке форма? Це засіб, який дозволяє відвідувачеві взаємодіяти з вашим сайтом: відправляти сайту дані, визначені вами у формі. Типовими прикладами форм є форми входу на сайт, пошуку по сайту, відправки повідомлення адміністратору сайту, реєстрації на сайті, опитування. Все перераховане і багато іншого робиться за допомогою форм HTML. Форми в HTML можуть складатися з текстових полів і текстових областей, прапорців і перемикачів, а також списків, що розкриваються. Все це - елементи форми. Кожен елемент служить для того, щоб передати якесь значення сайту. Щоб створити форму, слід скористатися парним блоковим тегом form. Все, що укладено в цьому тезі, буде вважатися формою. Перебувати в ньому можуть не тільки прапорці, текстові області і інші елементи форми, а й будь-які інші теги мови HTML. Звичайно, значення всіх тегів, які не є елементами форми, не будуть передаватися сайту при відправці форми. У першу чергу тому, що тегам, які не є

елементами форми, не можна присвоювати значення. Ці теги використовуються всередині форми лише для того, щоб надати формі потрібний дизайн:

Форма і її елементи:

<form method = "get" action = "">
<h3> Приклад форми </ h3>
 Це вже наша форма </ div>
</form>

Атрибут action. Кожна форма служить для того, щоб передавати значення на сайт. Але куди саме? За замовчуванням значення передаються на ту ж сторінку, де знаходиться форма, тобто після відправки форми в браузері відкривається та ж сторінка. Але за допомогою атрибута action можна змінити поведінку за замовчуванням. Якщо ви вкажете в значенні даного атрибута абсолютну або відносну адресу, то після відправки форми буде відкрита зазначена в значенні веб-сторінка. Бажано завжди задавати значення атрибута action. Навіть якщо форма повинна передати значення на ту ж сторінку, де вона розташована. У цьому випадку в якості значення атрибута action вкажіть порожній рядок: action = "".

Атрибут method. У тезі form також часто задається атрибут method. За його допомогою можна визначити спосіб, яким будуть передані на сайт значення, вибрані користувачем у формі. Найбільш популярними є два способи передачі значень сайту:

- get - передача значень в адресі веб-сторінки;
- post - у заголовку запиту до веб-сторінки.

Практично всі елементи форми створюються за допомогою тега input. І тільки від атрибута type цього одинарного тега залежить, який саме елемент форми буде створено:

<input type = "значення" />

Крім атрибута type, у тега input є ще кілька обов'язкових атрибутів. Тегу input слід додати атрибут name. Значення цього атрибута визначає назву змінної, в яку при відправці форми на сторінку буде передано обране в даному елементі

форми значення. Тегу input слід також додати атрибут value. Значення даного атрибута визначає саме значення, яке при відправці форми буде передано в змінній, зазначеної атрибутом name. При необхідності тегу input можна привласнити атрибут readonly зі значенням readonly. При наявності даного атрибута користувачу буде заборонено змінювати значення за замовчуванням, вказане вами в тезі input. У цьому випадку значення елемента форми можна буде змінити тільки за допомогою JavaScript. Замість атрибута readonly можна використовувати атрибут disabled зі значенням disabled. У цьому випадку елемент форми не тільки стає недоступним для зміни, а й забарвлюється в сірий колір. Крім того, значення, вказане в цьому елементі форми, не буде віддане на сервер при відправці форми. Наявність інших атрибутів залежить від типу створюваного елемента форми.

Кнопка відправки форми (submit). Щоб створити кнопку відправки форми, потрібно присвоїти атрибуту type значення submit:

<input type = "submit" value = "Відправити" name = "form_submit" />

Як показує практика, найчастіше застосовується такий елемент форми, як текстове поле. З його допомогою відвідувач може передати на сайт будь-який рядок тексту. Текстове поле створюється за допомогою значення text атрибута type: <input type = "text" name = "form_name" value = "ваше ім'я" />

Форма і її елементи. Атрибут value текстового поля призначений для того, щоб зберігати значення, введене користувачем в це поле. Але поки користувач нічого не ввів, атрибут value може містити в собі значення за замовчуванням, тобто ті значення, які ви вказали в атрибуті value при верстці. При цьому значення за замовчуванням буде відображатися в текстовому полі. При необхідності можна обмежити кількість символів, які відвідувач може ввести в текстовому полі. Для цього призначений атрибут maxlength, значення якого і є максимально дозволеним кількістю символів.

Поле введення пароля (password). Різновидом текстового поля є поле для введення пароля. Воно підтримує ті ж атрибути, що і текстове поле. Але весь текст, який відвідувач введе в даному полі, буде відображатися у вигляді зірочок

Отже, для створення поля введення пароля призначене значення password атрибута type:

<input type = "password" name = "form_pass" value = ""
maxlength = "8" />[2].

CSS працює з HTML, але не має до HTML ніякого відношення. Це абсолютно інша мова. HTML структурує документ, впорядковуючи інформацію в заголовки, абзаци, марковані списки і т.д., в той час як CSS тісно взаємодіє з браузером, щоб оформлення HTML-документа мало досконалий вигляд. Наприклад, ви могли б використовувати HTML, щоб перетворити фразу в заголовок, відокремлюючи його від змісту сторінки, але краще застосовувати CSS для форматування заголовка, скажімо, великим напівжирним червоним шрифтом з позиціонуванням на 50 пікселів від лівого краю вікна. В CSS це форматування тексту включає в себе стиль - правило, яке описує зовнішній вигляд конкретної частини веб-сторінки. А таблиця стилів (style sheet) є набором таких стилів. Можна також створювати стилі спеціально для роботи з зображеннями. Наприклад, за допомогою стилів можна вирівняти зображення по правому краю вебсторінки, помістити його в кольорову рамку, відокремити від навколишнього тексту на 50 пікселів.

Створивши стиль один раз, можна застосовувати його до текстових фрагментів, зображень, заголовкам і будь-яким іншим елементам сторінки скільки завгодно. Наприклад, ви можете вибрати абзац тексту і застосувати до нього стиль, що тут же змінює розмір, колір і шрифт тексту. Можна також зробити стилі для певних HTML-тегів так, щоб, наприклад, усі заголовки першого рівня (теги <h1>) на вашому сайті були відображені в однаковому стилі, незалежно від того, де вони розміщені[3].

Зрозуміло, CSS-стилі не можуть бути написані на звичайній мові, як, наприклад, попередній абзац. У них є власна мова. Зокрема, щоб встановити червоний колір і розмір шрифту 1,5 em для всіх абзаців на веб-сторінці, потрібно написати наступне:

p {color: red; font-size: 1.5em; }

Цей стиль ніби каже браузеру: «Розфарбуй текст всіх абзаців веб-сторінки, укладених в елемент p, червоним кольором і встанови розмір шрифту рівним 1,5 em (em - одиниця виміру розміру шрифту тексту в браузері). Будь-який стиль, навіть найпростіший, містить кілька елементів.

Селектор повідомляє браузеру, до якого елементу або елементів веб-сторінки застосовується стиль: до заголовку, абзацу, зображення чи посилання. Завдяки великій різноманітності селекторів, пропонованих мовою CSS, і невеликого творчого потенціалу можна майстерно форматувати веб-сторінки.

Блок оголошення. Код, розташований відразу за селектором, містить всі команди форматування, які можна застосувати до цього селектору. Блок починається з відкриваючої фігурної дужки { і закінчується закриваючою }.

Оголошення. Між відкриваючою і закриваючою фігурними дужками блоку оголошення можна додати одне або кілька оголошень - команд форматування. Кожне оголошення має дві частини - властивість і значення. Двокрапка відокремлює ім'я властивості від його значення, і все оголошення закінчується крапкою з комою.

Властивість. Каскадні таблиці стилів пропонують великий вибір команд форматування, званих властивостями. Властивість є слово або кілька написаних через дефіс слів, що визначають конкретний стиль. У більшості властивостей є відповідні прості для розуміння імена, такі як font-size, margin-top, background-color і т.д. (В перекладі з англійської: розмір шрифту, верхній відступ, колір фону)[4].

JavaScript - мова сценаріїв (скриптів), що застосовують в основному для створення інтерактивних елементів на Web-сторінках. ЇЇ можна використовувати для перевірки вірності заповнення форм, створення меню, зміни зображень або для інших задач, що можна зробити на Web-сторінках. Якщо подивитись на Google Maps чи службу GMail компанії Google, то можна зрозуміти, на що мова JavaScript здатна сьогодні. Оскільки JavaScript на сьогодні є єдиною мовою сценаріїв, яку можуть підтримувати усі основні браузери Web (Internet Explorer,

Safari, Firefox, Google Chrome, Opera і т.д.), то вона використовується достатньо широко.

Код JavaScript виконується Web-браузерами клієнтів, і в такому випадку він називається сценарієм на боці клієнта. Але код JavaScript можна виконувати також і на Web-сервері для формування документів HTML, виконуючи тим самим сценарій на стороні сервера. Хоча й використання JavaScript зазвичай обмежується сценаріями на боці клієнта, але він також є дуже могутньою серверною мовою.

При створенні коду JavaScript фактично потрібен тільки текстовий редактор та Web-браузер. По-перше, необхідно дізнатися, як додавати сценарій JavaScript на сторінку HTML. Це робиться одним з двох способів: помістити теги Script на Web-сторінці та розташувати код JavaScript усередині цих тегів, чи помістити увесь код JavaScript до окремого файлу і зв'язатися з ним за допомогою тега Script. Будь-який із цих методів цілком допустимий, але вони мають різні призначення. Якщо є невеличкий код, що буде використовуватися лише на одній сторінці, то його розміщення між тегами Script буде гарним рішенням. Але якщо є великий фрагмент коду, що буде використовуватися на кількох сторінках, то, напевно, доцільніше помістити цей JavaScript код до окремого файлу і з'єднатися із ним. Це робиться для того, аби не було потрібно завантажувати цей код кожного разу під час відвідування абсолютно різних сторінок. Код завантажується лише один раз, і браузер збереже його для подальшого використання[5].

Для виклику цієї мови його код поміщається між HTML-тегами <script> і </script>. Типовий документ Hello World, створений на HTML із застосуванням JavaScript може мати показаний нижче вигляд: фраза Hello World, яка відобра-жається за допомогою JavaScript

<html>

<head> <title> Hello World </ title> </ head>

<body>

<script type = "text / javascript"> document.write ("Hello World")

```
</script>
<noscript>
Ваш браузер не підтримує JavaScript, або його підтримка відключена
</noscript>
</body>
</html>
```

Всередині тегів <script> знаходиться всього один рядок коду JavaScript, в якому використовується команда document.write, що є еквівалентом PHP-команди echo або команди print. Як і було очікувано, вона просто виводить наданий їй рядок в поточний документ при його відображенні на екрані. Можна було також помітити, що, на відміну від PHP, в цій команді отсутствует замикаюча крапка з комою (;). Причина в тому, що в JavaScript дією, еквівалентною дії крапки з комою, є символ нового рядка. Проте, якщо буде потрібно розмістити в одному рядку більше однієї інструкції, то після кожної інструкції, крім останньої, потрібно поставити крапку з комою.

Сценарій можна вставити не тільки в тіло документа, а й і у розділ <head>, який є ідеальним місцем, якщо потрібно виконати сценарій при завантаженні сторінки. Присутність в цьому розділі якогось дуже важливого коду і функцій забезпечує також їх негайну готовність до використання в інших розділах, що мають сценарії, в тому документі, який залежить від їх застосування. Іншою причиною для вставки сценарію в заголовок документа може бути здатність JavaScript записувати в розділ <head> такі елементи, як мета-теги, оскільки те місце, куди вставляється сценарій, стає за замовчуванням частиною документа, куди він здійснює виведення інформації[6].

PHP - мова програмування, яка призначена для інтерактивного створення веб-сторінок на комп'ютері, який називається веб-сервером. На відміну від HTML, коли веб-броузер генерує сторінку на основі тегів і розмітки, PHP-код виповнюється між запитаною сторінкою і веб-сервером, додаючи і змінюючи основний код HTML. Мова PHP спрощує розробку веб-сторінок, оскільки платформа PHP містить весь необхідний програмний код. Це означає, що вам не

треба винаходити велосипед щоразу, коли буде потрібно написати програму на мові PHP; вся веб-функціональність вже включена в нього. Хоча PHP прекрасно підходить для створення веб додатків, зберіганням інформації сам він не займається. Розробники сценаріїв на мові програмування PHP зазвичай беруть базу даних MySQL, яка і служить діловодом для користувача інформації, що обробляється PHP.

СУБД MySQL автоматизує більшу частину завдань, пов'язаних зі зберіганням і витягом інформації користувача на основі заданих вами критеріїв. Звернемося до прикладу з сайтом Amazon.com: пропоновані сайтом рекомендації користувачеві засновані на інформації про його попередні замовлення, що збережена у базі даних. Організувати доступ до MySQL з PHP дуже легко, і вони прекрасно працюють разом. Додаткова перевага у тому, що PHP та MySQL можуть працювати на комп'ютерах різних типів, керованих різними операційними системами, в тому числі Mac OS X, Windows і Linux.

Ось кілька причин вважати спільне використання PHP та MySQL цілком природним рішенням:

PHP і MySQL прекрасно працюють разом. При розробці PHP і MySQL взаємно враховувалися їх особливості, тому їм легко працювати разом. Програмні інтерфейси між ними логічно пов'язані. Створюючи інтерфейси PHP і MySQL, розробники спочатку враховували можливість спільної роботи.

PHP і MySQL - програмні продукти, що мають відкритий вихідний код. Оскільки і PHP, і MySQL є проектами з відкритим кодом, вони можуть використовуватися без будь-яких обмежень. Клієнтські бібліотеки MySQL більше не прив'язані до PHP. Досвідчені користувачі можуть вносити зміни в початкові тексти, змінюючи таким чином порядок роботи мови і програм.

PHP і MySQL мають підтримку у вигляді спільнот. В Мережі є активні спільноти, в які ви можете вступити, а їх учасники допоможуть вам знаходити відповіді на питання. Крім того, при необхідності ви можете звертатися до платних фахівців з MySQL.

PHP і MySQL працюють швидко. Більш швидка робота - наслідок простоти і доцільності їх пристрою.

PHP і MySQL не дадуть потонути в незначних деталях. Не потрібно знати всі найдрібніші подробиці взаємодії мови PHP з базами даних MySQL, оскільки є стандартний інтерфейс виклику процедур MySQL з PHP.

Мова PHP народилась з потреби розробляти і підтримувати веб-сайти, які мають динамічну клієнт-серверну функціональність. У 1994 році Расмус Лердорф розробив набір сценаріїв з відкритим вихідним кодом на мові Perl, які згодом були переписані на мові C і перетворилися в те, чим є сучасна мова PHP. У 1998 році вийшла третя версія PHP, і тепер цей інструмент створення веб-додатків може конкурувати з аналогічними продуктами, наприклад Active Server Pages (ASP) компанії Microsoft і Java Server Pages (JSP) компанії Sun. PHP - інтерпретована, а не компільована мову.

HTML-форми дають змогу передавати уведені користувачем дані до серверу, де їх можна ще додатково обробляти. Перевірку і обробку даних форми можна виконувати із застосуванням більшості понять мови PHP.

Обслуговування форм здійснюється у два етапи. Спочатку форма має бути представлена користувачеві, який заповнятиме її своїми даними та потім відправить на сервер. Кожна форма має цільову вебсторінку, що має бути завантажена для обробки даних, які відправив користувач. Найчастіше це є той самий файл сценарію, що генерує форму. У такому випадку PHP код просто перевірить наявність даних у формі, аби визначити, чи викликати йому знову сценарій для створення форми чи просто почати обробку отриманих даних.

Операція «пошук» у базі даних - необхідна дія для переважної більшості різноманітних додатків, як то пошук повідомлень у форумі, користувачів або блогу. У будь-якому випадку ця операція може значно полегшити життя користувачів. На рівні бази даних існує багато різних способів виконувати пошук і повертати отримані результати[7].

SQL - мова структурованих запитів - є стандартною мовою керування базами даних. Його прототип був розроблений фірмою IBM на основі ідей,

викладених у статті Е.Ф. Кодда "Реляційна модель даних для великих банків даних загального користування". Трохи пізніше появи прототипу IBM, в 1979 році, на ринку з'явився перший продукт SQL під назвою ORACLE, який був випущений компанією Relational Software, Incorporated (згодом перейменованої в Oracle Corporation). Сьогодні ця компанія є одним з видатних лідерів в області реалізації технологій реляційних баз даних. SQL можна вимовляти або по буквах - S-Q-L, або як "сіквел" (sequel) - обидва вимови прийнятні.

База даних - це деяка сукупність даних. Деякі вважають за краще уявляти собі базу даних як якийсь організований механізм, здатний зберігати інформацію, за допомогою якого користувач може цю інформацію витягти ефективним і корисним для себе чином.

Реляційна база даних - це така база даних, яка розділена на логічно цілісні сегменти, що звані таблицями, і всередині бази даних таблиці пов'язані одна з одною. Реляційна база даних дозволяє розділяти дані на логічні дрібніші і більш керовані сегменти, що забезпечить оптимальне представлення даних та можливість організації декількох рівнів доступу до даних.

У минулому комп'ютерна індустрія ґрунтувалася на використанні мейнфреймів - великих і потужних комп'ютерів із значними можливостями для зберігання і обробки даних. Користувачі мали можливість спілкування з мейнфреймами за допомогою "тупих" терміналів. Ці термінали не мали своїх власних "інтелектуальних" можливостей і покладалися виключно на обчислювальні можливості, пам'ять і носії інформації мейнфрейма. Кожен термінал мав свою лінію обміну даними з мейнфреймами. Устаткування мейнфреймів цілком справлялося зі своїми завданнями і відповідало вимогам бізнесу того часу, але прийшов час для нової, значно більш прогресивної технології - моделі клієнт / сервер.

Сеанс SQL почнеться в момент підключення до бази даних. Для цього використовують команда CONNECT. За допомогою команди CONNECT можливо або здійснювати підключення до бази даних, або змінювати характер вже встановлених підключень. Наприклад, якщо підключитись до бази даних під

ім'ям USER01, ви зможете потім використовувати команду CONNECT, аби підключитися до тої ж самої бази даних під ім'ям USER02. При цьому неявно припиниться сеанс SQL для користувача USER01.

У разі спроби підключитися до бази даних буде автоматично отримано запит на введення пароля, що є відповідним введеному імені користувача.

Сеанс SQL припиниться при відключенні користувача від бази даних. Для відключення користувача від БД використовують команду DISCONNECT. Після відключення від бази даних ще є можливість користуватися програмними засобами зв'язку з базою даних, але сам зв'язок з базою даних припиниться. Якщо для розриву зв'язку використати оператор EXIT - припиниться не тільки ваш сеанс SQL, але і закриється програма, за допомогою якої був здійснений доступ до бази даних[8].

Поняття «веб-сервер» може відноситись як до апаратної частини, так і до програмного забезпечення. Або навіть до двох частин, які працюють сумісно.

З точки зору "апаратної", "веб-сервер" - це комп'ютер, що зберігає файли сайту (HTML документи, CSS стилі, JavaScript файли, зображення та інші) і надає доступ до них пристрою кінцевого користувача (веб-браузер тощо). Він має бути підключений до мережі Інтернет і зможе бути доступним через доменне ім'я, подібне google.com.

З точки зору ПЗ, веб-сервер складається з декількох компонентів, що контролюють доступ користувачів до розміщених на сервері файлів, насамперед - це HTTP-сервер. HTTP-сервер - це частина ПЗ, яка розуміє URL'и (веб-адреси) і HTTP (протокол, який браузер використовує для перегляду веб-сторінок).

На базовому рівні, коли браузеру потрібен файл, що розміщується на веб-сервері, браузер запитує його за допомогою HTTP-протоколу. Коли запит досягає потрібного веб-сервера (апаратна частина), сервер HTTP (ПЗ) приймає запит, знаходить запитуваний документ (якщо немає, то повідомляє про помилку 404) і відправляє назад, також через HTTP.

Для того, щоб веб-сайт був опублікований, необхідний статичний або динамічний веб-сервер.

Статичний веб-сервер (стек) складається з комп'ютера (апаратне забезпечення) з HTTP-сервером (ПЗ). Це називається статикою, тому що сервер посилає розміщені файли в браузер такими, як вони є.

Динамічний веб-сервер створений зі статичного веб-сервера та додаткового ПЗ, найчастіше сервера додатку і бази даних. Ми називаємо його «динамічним», тому що сервер додатків змінює вихідні файли перед відправкою в ваш браузер по HTTP.

Наприклад, щоб отримати підсумкову сторінки, яка переглядається в браузері, сервер може заповнити HTML-шаблон даними з бази даних. Такі сайти, як facebook або wikipedia, складаються з мільйонів або навіть мільярдів веб-сторінок, але вони не є реальними HTML документами – це лише декілька HTML-шаблонів та гігантські бази даних. Ця структура сильно спрощує і прискорює керування веб-додатками і відображення контенту.

Спершу, веб-сервер має містити файли веб-сайту, а саме: усі HTML-документи і пов'язані з ними ресурси, включаючи зображення, JavaScript скрипти, CSS стилі, відео та шрифти. Технічно, можна розмістити всі ці файли на своєму комп'ютері, але набагато зручніше зберігати їх на виділеному веб-сервері, який:

- постійно запущений і активний
- має постійний доступ до Інтернету
- має статичну IP адресу (не кожен провайдер надає статичну IP-адресу для домашніх підключень)
- обслуговується третьою, сторонньою компанією[9].

Якщо розглянути деякий веб-додаток, то можна припустити, що він містить цілий ряд компонентів, які, якщо будуть зламані, здатні завдати істотної шкоди.

Шкода користувачам: отримання доступу до електронної пошти, паролів, персональних даних, реквізитів банківських карт, ділових таємниць, списків контактів, історії транзакцій і глибоко охоронюваних секретів. Витік цих даних шкодить користувачам (приватним особам і компаніям). Нашкодити можуть

також веб-додатки, неправильно застосовуючі подібні дані, і вузли, які використовують довіру користувачів в своїх інтересах.

Шкода самої компанії: через заподіяння користувачам шкоди погіршується репутація, доводиться виплачувати компенсації, втрачається важлива ділова інформація, виникають додаткові витрати - на інфраструктуру, поліпшення безпеки, ліквідацію наслідків, судові витрати і т.д.

Перевірка необхідна при кожній передачі даних в новий контекст. Це стосується і до перевірки, виконуваної поза самого веб-додатки. До подібних засобів відноситься перевірка чи інші обмеження, що застосовуються до HTML-форм в браузері.

HTML-форми вміють накладати обмеження на дані, що вводяться. Можна обмежувати вибір за допомогою списку з фіксованих пунктів, задавати мінімальні і максимальні значення, а також обмежувати довжину тексту. Можливості HTML 5 ще ширше. Браузери можуть перевіряти URL і адреси пошти, контролювати дати, числа і діапазони (хоча підтримка останніх двох досить умовна). Також браузери здатні перевіряти вводяться дані за допомогою регулярних виразів JavaScript, включених в атрибут шаблону.

З усім цим достатком засобів контролю не можна забувати: їх призначення - поліпшити зручність вашого застосування. Будь-який зловмисник здатний створити форму, яка не буде містити обмежень з вашої вихідної форми. Можна навіть створити HTTP-клієнт для автоматизованого заповнення форм.

Інший приклад зовнішніх коштів перевірки - отримання даних від сторонніх API, наприклад Twitter'a. Ця соціальна мережа має гарну репутацію, і їй зазвичай довіряють без питань. Але не варто довіряти навіть Twitter'у. При компрометації в його відповідях з'являться небезпечні дані, до появи яких ми не будемо готові. Тому навіть тут застосовуйте власну перевірку, щоб не опинитися беззахисними в разі чого. Там, де ми впевнені в зовнішніх засобах перевірки, зручно відстежувати уразливості. Наприклад, якщо HTML-форма встановлює обмеження на максимальну довжину і ми отримуємо вхідні дані, чий розмір досяг межі, то логічно припустити, що це користувач намагається обійти

перевірку. Таким чином ми можемо реєструвати проломи в зовнішніх засобах і робити подальші дії проти потенційних атак, обмежуючи доступ або кількість запитів[10].

Одна з проблем, яка виникає перед WEB-ресурсами, що мають персональні кабінети - атака перебором. Так, простий перебір всіх варіантів пароля для конкретного акаунту. Така атака може сильно навантажити ресурс. До того ж, якщо контролю складності пароля користувача при реєстрації немає, вона може виявитися ще й успішною.

Найчастіше, питання вирішується відносно просто. Якщо користувач ввів кілька разів неправильно пароль, його акаунт блокується на якийсь час. Альтернативне рішення - виводити капчу. Відразу, або після кількох невдалих спроб. Ну, і не забудемо про 2F авторизацію, яка майже невразлива.

Тимчасове блокування - акаунт користувача тимчасово заблокований і він не може потрапити в систему. Реальний користувач в період атаки відчуває душевний біль і муки. Він не може потрапити в систему. І швидше за все навантажує технічну підтримку. А найцікавіше, що, можливо, в цьому і є мета атакуючого.

Капча - відносно непогане і ефективне рішення. Правда доставляє незручність користувачеві, вимагаючи вводити щось там додатково. Досить "неприємно" вбудовувати в дизайн. В залежності від реалізації, може бути схильна до DoS атаки.

2F авторизація є дієвим засобом, але не деяких ресурсах вводити такий захід безпеки є занадто ресурсозатратним[11].

Технологія CAPTCHA - абревіатура слів англійською: "Completely Automated Public Turing Test to Tell Computers and Humans Apart", що перекладається як «Повністю автоматичний тест Тьюринга щоб розрізнити комп'ютери та людей». Цей тест, запропонований британським вченим Аланом Тьюрингом у статті 1950 року «Computing machinery and intelligence», тобто «обчислювальні машини і розум» для перевірки, чи є комп'ютер розумним у людському розумінні цього слова. Людина (суддя) переписується звичайною

мовою із двома співрозмовниками, один з яких – комп'ютер, а інший - людина. Якщо суддя не зможе однозначно визначити, хто є реальною людиною, а хто комп'ютером, то це означає, що комп'ютер пройшов тест. Передбачається, що кожний зі співрозмовників має мету, щоб саме його визнали людиною.

Найвідоміша реалізація CAPTCHA - це буквено-цифровий тест. Його суть полягає у наступному: при заповненні форми людині відображається картинка з деяким випадковим набором букв та цифр, а поруч із цією картинкою текстове поле в яке необхідно ввести символи, що зображені на картинці. Якщо введені в текстовому полі символи та символи на картинці збігаються, тоді заповнена форма приймається до обробки, а якщо ні - генерується нова картинка і людину просять увести символи з картинки ще раз. Це дуже просте завдання для реальної людини, але для бота воно досить важке[12].

1.2. Вразливості форм авторизації веб-сайтів

Для комплексного розуміння того, як вирішити задачу підвищення безпеки форм авторизації, необхідно класифікувати та проаналізувати існуючі вразливості веб-сайтів. Розіб'ємо відомі методи атак на веб-сайти на основні пункти.

Аутентифікація (Authentication). Це стосується атак, що спрямовані на методи перевірки веб-додатками ідентифікаторів користувачів, програм або служб. Далі будуть описані атаки, спрямовані на обхід чи експлуатацію вразливостей в механізмах, які реалізують аутентифікацію веб-серверів. Аутентифікація використовує як мінімум один з трьох механізмів (факторів): “щось, що:

- ми маємо
- ми знаємо
- ми є”

Атака підбором (BruteForce). Підбір - запрограмований процес проб і помилок, що використовується для того, аби вгадати логін користувача, пароль, ключ шифрування, номер кредитної картки, і т.д. Існує безліч систем, що дозволяють користувачам використовувати ненадійні паролі чи ключі шифрування, а користувачі часто обирають легко вгадувані або ті, що містяться в загальнодоступних словниках паролі.

Зловживаючи такою ситуацією, зловмисник має змогу скористатися словником та спробувати використати десятки тисяч або навіть мільйонів комбінацій символів з нього, в якості потенційного пароля. Якщо один з випробуваних паролів дозволяє отримати доступ до системи, атаку можна вважати успішною і атакуючий зможе використовувати обліковий запис та отримати доступ до особистих даних користувача. Така техніка проб і помилок може використовуватись для підбору ключів шифрування. У разі використання ключів недостатньою довжини, зловмисник може отримати актуальний ключ, просто перебравши усі можливі комбінації[13].

Існує два види перебору: прямий та зворотний. При зворотному перебирають різні логіни користувачів, а пароль залишається тим самим. При прямому підборі використовують різні варіанти пароля для одного й того ж логіну користувача. В системах, що мають мільйони облікових записів, ймовірність використання різними користувачами одного пароля дуже висока. Не дивлячись на популярність і високу ефективність, підбір інколи може зайняти кілька годин, днів, тижнів або навіть років[14].

Недостатня аутентифікація. Це вразливість, що виникає коли Web-сервер дозволяє атакуючому отримати доступ до важливих даних або функцій сервера без достатньої аутентифікації. Інтерфейси адміністрування через веб – є яскравим прикладом таких систем.

В залежності від особливостей додатку, подібні компоненти не повинні бути доступні без належної аутентифікації. Деякі ресурси, щоб не використовувати аутентифікацію "ховаються" за декякою адресою, що не зазначена на основних сторінках сервера чи яких-небудь інших

загальнодоступних ресурсах. Проте, подібний підхід є не більш ніж "безпека через приховування". Важливо розуміти, що, не зважаючи на те, що потенційний зловмисник не знає адресу сторінки, вона у будь-якому випадку буде доступна через Web.

Потрібний URL можна знайти підбором типових назв файлів і директорій (наприклад /admin/), із використанням повідомлень щодо помилок, журналів перехресних посилань або шляхом простого читання документації. Такі ресурси необхідно захищати адекватно важливості їх вмісту та функціональних можливостей.

Велика кількість Web-додатків за вмовчанням для адміністративного доступу використовують посилання в кореневій директорії сервера (/admin/). Зазвичай посилання на цю сторінку не з'являється у вмісті сервера, проте ця сторінка доступна за допомогою звичайного браузера. Оскільки користувач чи розробник припускає, що ніхто не скористається цією сторінкою, так як посилання на неї відсутнє, часто реалізацією аутентифікації нехтують. І для отримання контролю над сервером зловмисникові досить зайти на цю сторінку.

Небезпечне відновлення забутих паролів. Це вразливість, що виникає, коли Web-сервер може дозволяти атакуючому несанкціоновано отримувати, редагувати або відновлювати паролі, які належать іншим користувачам.

Часто аутентифікація на Web-сервері вимагає від користувача запам'ятовування пароля або парольної фрази. Лише користувач повинен знати пароль, причому пам'ятати його абсолютно точно. З часом пароль забувається. Ситуація може ускладнюватись, оскільки середній користувач відвідує близько 20-30 сайтів, які потребують введення пароля. Тож, функція відновлення пароля є дуже важливою складовою сервісу, що надають веб-ресурси.

Прикладом реалізації такої функції є використання "секретного питання", на яке вказують відповідь в процесі реєстрації. Питання або можна вибрати зі списку або вигадується самим користувачем[15].

Ще один механізм дозволяє користувачам вказати так звану "підказку", що зможе допомгти йому згадати свій пароль. Інші способи вимагають від

користувача вказати частину особистих даних, таких як ІПН, поштовий індекс, домашню адресу і т.д., що згодом будуть використовуватися для встановлення особи. Після того як користувач довів свою ідентичність, система відображає новий пароль або надішле його поштою.

Вразливості пов'язані з недостатньою перевіркою при відновленні пароля виникають, коли атакуючий отримує можливість використовується механізм. Це трапляється, коли інформацію, використовувану 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 вгадується відповіді на секретне питання.

Приклади слабких методів відновлення паролів:

Перевірка інформації. Багато серверів вимагають від користувача вказувати його email в комбінації з номером телефону і домашньою адресою. Така інформація може бути дуже легко отримана з мережевих довідників. В результаті цього, дані, що використовуються для перевірок, не є великим секретом. Окрім того, ця інформація також може бути отримана зловмисником за допомогою інших методів, таких як міжсайтове виконання сценаріїв або фішинг (phishing).

Парольні підказки. Сервер, який використовує підказки з метою полегшення запам'ятовування паролів, може бути атакований, оскільки підказки допомагають у підборі паролів. Користувач може використати стійкий пароль, наприклад, "221290King" з підказкою: "день народж + улюбл письменник". Атакуючий зможе здогадатися, що пароль користувача складається з дати народження та імені улюбленого автора користувача. Це допоможе сформувати відносно короткий словник для атаки методом перебору.

Таємне питання і відповідь. Припустимо, відповідь користувача "Одеса", а секретне питання "Місто народження". Зловмисник може обмежити словник для підбору секретного відповіді назвами міст. Більше того, якщо атакуючий володіє деякою інформацією щодо користувача, то дізнатися його місце народження дуже легко[16].

Авторизація (Authorization). Цей пункт присвячений атакам, спрямованим на методи, які використовуються веб-сервером для визначення того, чи має користувач, додаток або служба необхідні для вчинення дії дозволу. Багато веб-сайтів дозволяють лише певним користувачам отримувати доступ до деякого вмісту або функцій програми. Доступ іншим користувачам повинен бути обмежений. Використовуючи різні техніки, зловмисник може підвищити свої привілеї і отримати доступ до захищених ресурсів.

Передбачення значення ідентифікатора сесії (Credential / Session Prediction). Ця методика дозволяє перехоплювати сесії інших користувачів. Подібні атаки виконуються вгадування унікального ідентифікатора сесії користувача або шляхом передбачення. Ця атака (так само як і перехоплення сесії (Session Hijacking)) у разі успіху дозволить зловмисникові надіслати запит веб-серверу з правами скомпрометованого користувача. Дизайн багатьох серверів передбачає аутентифікацію користувача при першому зверненні та подальше відстеження його сесії. Для цього користувач вказує комбінацію логіну та пароля. Замість повторної передачі імені користувача і пароля при кожній транзакції, Web-сервер генерує унікальний ідентифікатор, який присвоюється сесії користувача. Наступні запити користувача до сервера містять ідентифікатор сесії як доказ того, що аутентифікація була успішно пройдена. Якщо атакуючий може передбачити або вгадати значення ідентифікатора іншого користувача, це може бути використано для проведення атаки[17].

Процедура авторизації визначає, які дії може здійснювати користувач, служба або додаток. Правильно побудовані правила доступу повинні обмежувати дії користувача відповідно до політики безпеки. Доступ до важливих ресурсів сайту повинен бути дозволений тільки адміністраторам[18].

Відсутність таймаута сесії (Insufficient Session Expiration). У разі якщо для ідентифікатора сесії або облікових даних не передбачений таймаут 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. Таким чином, конфіденційність кожного ідентифікатора повинна бути забезпечена, щоб запобігти множинний доступ користувачів з одним обліковим записом.

В іншій ситуації, якщо користувач отримує доступ до сервера з публічного комп'ютера (бібліотека, Internet-кафе і т.д.) відсутність таймауту сесії може дозволити зловмиснику скористатися історією браузера для перегляду сторінок пользователя. Велике значення таймаута збільшує шанси підбору чинного ідентифікатора. Крім того, збільшення цього параметра веде до збільшення одночасно відкритих сесій, що ще більше підвищує ймовірність успішного підбору[19].

Фіксація сесії (Session Fixation). Використовуючи даний клас атак, 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 моменту, коли користувач увійде в систему. Після входу користувача, зловмисник використовує ідентифікатор сесії для отримання доступу до системи від імені користувача.

Можна виділити два типи систем управління сесіями на основі ідентифікаторів. Перший з них, "запускає", дозволяє браузеру вказувати будь-який ідентифікатор. Системи другого "суворого" типу обробляють тільки ідентифікатори, згенеровані сервером. Якщо використовуються "дозволяють" системи, зловмисник може вибрати будь-який ідентифікатор сесії. У випадку зі "строгими" серверами зловмисникові доводиться підтримувати "сесію-заглушку" і періодично з'єднуватися з сервером для уникнення закриття сесії з таймаут.

На відміну від крадіжки ідентифікатора, фіксація сесії надає зловмисникові набагато більший простір для творчості. Це пов'язано з тим, що активна фаза атаки відбувається перед входом користувача в систему.

1.3. Висновки до першого розділу

В ході виконання першого розділу дипломної роботи було виконано огляд світової літератури щодо фундаментальних технологій, на яких працюють сучасні вебсайти. Теоретична база технологій HTML, CSS, PHP, SQL дуже потужно розвинена і не потребує допрацювань. Були розглянуті твори, щодо інформаційної безпеки веб додатків. Виокремлено основні принципи інформаційної безпеки. Особливої уваги потребують вразливості веб-сайтів пов'язані з авторизацією та формами. Необхідно дослідити можливість підвищення безпеки особистих даних користувачів веб-сайтів.

Розділ 2. РОЗРОБКА МЕТОДИКИ ВИРІШЕННЯ ЗАДАЧІ ПІДВИЩЕННЯ БЕЗПЕКИ ФОРМ АВТОРИЗАЦІЇ.

2.1. Атака повним перебором

Більшість користувачів вважають, що якщо вони застосовують довгий пароль або складний алгоритм кодування, то все буде безпечно. Спочату чергу треба запам'ятати, що немає ідеальних паролів; вся справа лише в часі, який знадобиться для їх злому. Наприклад, щоб зламати потужний криптографічний механізм, знадобиться 200 років, але злом все ж можливий, а з кожним днем цей час буде зменшуватися пропорційно збільшенню обчислювальної потужності комп'ютерів. Пароль, на зламання якого 10 років тому знадобилося б років 100, зараз можна зламати за тиждень. Якщо у хакера досить потужний комп'ютер для реалізації повного перебору цифр, букв і спеціальних символів, то паролю, зрештою, не встояти. Такий вид атаки іноді називається "метод грубої сили".

При цій атаці береться, наприклад, літера а й пробуються комбінації аа, ab, ас і так далі; потім ааа, aab, аас і так до тих пір, поки програма не підбере слово.

Варто відзначити, що іноді адміністратори самі полегшують завдання злому. Наприклад, визначаючи мінімум символів в паролі. Якщо хакер знає, що 6 символів - це мінімум, то початкової комбінацією буде аааааа. Навіщо пробувати всі одно-, дво-, три-, чотири-, п'ятисимвольні слова, якщо вони заборонені в системі?

З іншого боку, адміністратору доведеться зробити нелегкий вибір - встановити допустимий мінімум довжини слова і допомогти тим самим зловмисникові, або нічого не обмежувати - і нехай користувачі самі вибирають будь-яку довжину. Якщо взяти слово з чотирьох букв, то його можна розгадати дуже швидко. Напевно, краще все ж обмежувати мінімальну довжину паролів, тому що інакше користувачі візьмуть невеликі слова, що буде набагато гірше.

На головній арені битви з хакерами зараз знаходяться швидкість ЦПУ і час, необхідний для злому. Сучасні домашні комп'ютери мають обчислювальною потужністю центральних серверів, які використовувалися 10 років тому. Пам'ять постійно дешевшає, швидкість процесорів зростає, і з кожним днем паролі у всіх компаніях стають все більш легкими для злому.

Недостатня авторизація (Insufficient Authorization). 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.

Викрадений ідентифікатор може використовуватися для доступу до даних користувача або здійснення шахрайських транзакцій. Відсутність таймаута сесії значно збільшує ймовірність успіху деяких атак. Наприклад, зловмисник може отримати ідентифікатор сесії, використовуючи вразливість типу міжсайтового виконання сценаріїв або мережевий аналізатор. Хоча таймаут не здатен допомгти в разі, якщо ідентифікатор був використаний негайно, але обмеження часу дії допоможе у випадку пізніших спроб використання ідентифікатора.

Не варто забувати і про такий важливий аспект як розподілені атаки. Якщо хакеру потрібно швидко зламати пароль, йому зовсім не обов'язково купувати кілька швидких, тому й дорогих, комп'ютерів. Він може зламати кілька Web-вузлів з потужними системами і скористатися їх перевагами в швидкодії.

Уважно проаналізувавши тенденції розвитку, можна зробити висновок, що в найближчі кілька років компанії перейдуть на роботу з операційними системами, в які будуть вбудовані потужні механізми реєстрації і шифрування, отримають поширення одноразові паролі для аутентифікації або альтернативні методи реєстрації на кшталт тих, що пропонує біометрія.

Інтервал між зміною паролів повинен бути менше, ніж час, необхідний для їх злому. Таким чином, навіть якщо хтось методом повного перебору і дізнається пароль, то він вже не зможе ним скористатися. Якщо пароль можна зламати за 60

днів, значить змінювати його треба кожні 45 днів. На жаль, в більшості компаній все йде з точністю до навпаки. Їх паролі можна зламати менше ніж за 5 днів, а інтервал заміни близько дев'яти місяців. Тобто навіть якщо настирливому зловмисникові і знадобиться 3 місяці, щоб зламати пароль, у нього в запасі залишиться півроку. З огляду на сучасний стан мережевої безпеки, інтервал в 90 днів абсолютно неприйнятний.

Безумовно, плюси і мінуси є у кожного рішення. Припустимо, якщо зменшити інтервал зміни паролів з 12 місяців до 60 днів, з'явиться безліч проблем і труднощів, користувачі будуть вкрай незадоволені, і служба підтримки буде перевантажена боротьбою з користувачами, які записують де завгодно свої паролі. Найкраще все робити поступово, крок за кроком змінюючи умови використання паролів. Змінити інтервал з 12 місяців до 11, потім до 10 і так далі, до тих пір, поки не буде досягнутий нормальний показник.

Важливо детально пояснювати службовцям суть всіх вироблених змін. Найголовніший недолік зменшення часу зміни паролів в тому, що люди почнуть їх забувати, плутатися, і в кінцевому підсумку записувати. Ось тут допоможе передбачливість і уважність до своїх колег по роботі.

Атаки грубої сили відбуваються на ранніх стадіях ланцюга кіберзахисту, як правило, на етапах розвідки та проникнення. Зловмисникам потрібен доступ або точки входу в цілі, а методи грубої сили - це метод "встановіть і забудьте", щоб отримати такий доступ. Після того, як вони потрапляють у мережу, зловмисники можуть використовувати прийоми грубої сили для ескалації своїх привілеїв або для запуску атак зниження рівня шифрування.

Зловмисники також використовують атаки грубої сили для пошуку прихованих веб-сторінок. Приховані веб-сторінки - це веб-сайти, які існують в Інтернеті, але не пов’язані з іншими сторінками. Атака грубої сили перевіряє різні адреси, щоб перевірити, чи повертають вони дійсну веб-сторінку, і буде шукати сторінку, яку вони можуть використати. Такі речі, як уразливість програмного забезпечення в коді, який вони можуть використовувати для проникнення.

Види атаки грубої сили. Існує цілий ряд різних видів атаки грубою силою, кожен з яких має однакові цілі, детально описані вище.

Гібридні атаки грубої сили. Словникові атаки. Це одна з найпоширеніших форм атаки грубої сили, і для злому паролів використовується список слів у словнику. Інші типи атак можуть використовувати список часто використовуваних паролів. Наприклад, якщо ваш пароль - "пароль", бот грубої сили зможе зламати ваш пароль протягом декількох секунд.

Зворотний напад грубої сили. Зворотні атаки грубої сили не націлені на конкретне ім’я користувача, а замість цього використовують загальну групу паролів або індивідуальний пароль проти списку можливих імен користувачів.

Наповнення облікових даних. Коли зловмиснику відоме спарювання імені користувача та пароля, він може використовувати цю інформацію, щоб отримати доступ до кількох веб-сайтів та мережевих ресурсів. Наприклад, багато користувачів вибирають один і той же пароль для доступу до багатьох різних веб-сайтів заради простоти. Застосування заходів обережності, таких як використання двофакторної автентифікації та використання різних паролів для всіх різних мережевих ресурсів, може допомогти запобігти атакам грубої сили, які залежать від набору облікових даних.

Зупинимося ще на різновидах техніки повного перебору паролів - так званої атаки по словнику. Це метод, за допомогою якого можна розкрити осмислений пароль. Метод заснований на тому, що користувач для більш легкого запам'ятовування вибирає існуюче в деякій мові (словникове) слово. Якщо врахувати, що в будь-якій мові не більше 100.000 слів, очевидно, що повний перебір словникових слів станеться протягом невеликого проміжку часу.

Зараз широко поширені програми, що підбирають паролі на основі словникових слів. Тепер тільки безвідповідальний або ледачий користувач може зупинитися на осмисленому паролі. Нагадаємо, що, крім перевірки за словником, такі програми «вміють» змінювати регістри символів, «знають» розділові знаки, «здогадуються», що користувач може перевернути слово, склеїти два слова за допомогою розділового знака або цифри і т.п. перетворення.

Примітно, що сучасні розвинені засоби захисту від несанкціонованого доступу, які надають користувачу самостійно вибирати пароль для доступу, забезпечені модулями, які здійснюють перевірку обраного пароля на приналежність до такого роду словників і не допускають в такому випадку пароль до застосування.

Мета атаки грубої сили. Коли хакер зробить успішну спробу входу, що далі? Відповіддю може бути цілий ряд речей. Ось деякі з основних:

- Крадіжка або розголошення особистої інформації користувачів, знайденої в онлайн-акаунтах
- Збір комплектів облікових даних для продажу третім особам
- Подання себе власником облікового запису для поширення підробленого контенту або фішингових посилань
- Крадіжка системних ресурсів для використання в інших видах діяльності
- Дефект сайту через отримання доступу до облікового запису адміністратора
- Поширення шкідливого або спам-контенту або перенаправлення доменів на шкідливий контент

Як уже згадувалося, атаки методом "грубої сили" також можуть використовуватися для перевірки вразливостей в системі, тому вони не обов'язково шкідливі.

Приклади атак грубою силою. Атаки грубою силою відбуваються постійно, і є багато гучних прикладів. Ми, ймовірно, навіть не знаємо про багатьох минулих і триваючих атаках, але ось деякі з них, які виявилися в останні роки:

- Alibaba: Масова атака грубої сили 2016 року у популярний сайт електронної комерції торкнулася мільйони акаунтів.
- Magento: У березні 2018 року Magento повинен був попередити користувачів про те, що до 1000 адмін-панелей були зламані в результаті атак методом перебору.

- Північний ірландський парламент: Також в березні 2018 зловмисники перейшли на рахунки декількох членів парламенту Північної Ірландії.
- Вестмінстерський Парламент: Більш рання атака вдарила по Вестмінстерському парламенту в 2017 році, коли було зламано до 90 облікових записів електронної пошти.
- FireFox: На початку 2018 року було оголошено, що функція «майстер-пароля» в Firefox може бути легко зламана методом злому.

Це означає, що за останні дев'ять років облікові дані багатьох користувачів могли бути викрадені Незважаючи на те, що зловмисники часто використовують атаки грубої сили, вони вміли допомогти протестувати системи. Більш того, вони можуть запропонувати варіант резервного копіювання для відновлення пароля, якщо інші методи були вичерпані.

Як визначити атаку грубою силою. Нерідко користувачі отримують лист від постачальника послуг, в якому говориться, що хтось увійшов до вашого профілю з випадкового розташування. Якщо це відбувається, можливо, ви стали жертвою атаки грубої сили. В цьому випадку рекомендується змінити пароль негайно. Можливо, ви навіть захочете регулярно міняти свій пароль для конфіденційних облікових записів, на випадок, якщо ви стали жертвою непоміченою або незареєстрованої атаки методом підбору.

Якщо ви є адміністратором мережі, для безпеки вашого веб-сайту і ваших користувачів важливо стежити за ознаками грубої атаки, особливо успішною. Хоча купа невдалих входів в систему може бути від забудькуватого користувача, швидше за все, сайт знаходиться під атакою. Ось деякі ознаки, на які потрібно звернути увагу:

- Кілька невдалих спроб входу з однієї IP-адреси. Хоча це може бути результатом використання проксі-сервера великою організацією.
- Спроби входу в систему з кількома іменами користувачів з однієї IP-адреси. Знову ж таки, це може бути просто з великої організації.

- Кілька спроб входу в систему для одного імені користувача з різних IP-адрес. Це також може бути одна людина, що використовує проксі.
- Незвичайний шаблон невдалих спроб входу в систему, наприклад, по послідовному алфавітному або числовому шаблоном.
- Ненормальна величина смуги пропускання, яка використовується після успішної спроби входу в систему. Це може сигналізувати про напад, призначеному для крадіжки ресурсів.

З точки зору користувача може бути дуже важко дізнатися, чи був ваш обліковий запис зламаний шляхом злому. Якщо ви отримали повідомлення після того, як ваш обліковий запис був зламаний, найкраще перевірити свій обліковий запис на наявність змін, які не були внесені вами, і негайно змінити свій пароль.

З точки зору користувача, надійний пароль є обов'язковою умовою. Використання загального пароля або простого слова зі словника значно спростить потрапляння інструменту атаки методом грубої сили на правильний. Придумати надійний пароль може бути складно, але ось кілька порад:

- Довгі паролі краще, так як послідовний інструмент буде довше проходити через ітерації.
- Використання комбінації великих і малих літер, цифр і спеціальних символів зробить пароль надійнішим.
- Ніколи не використовуйте один і той же пароль для різних облікових записів, ви будете менш уразливі для певних типів атак ..

Звичайно, придумати і запам'ятати надійні паролі може бути складно, але є інструменти, які допоможуть вам. До них відносяться інструменти генерації паролів, інструменти для перевірки надійності паролів і інструменти менеджера паролів, такі як LastPass, KeePass, Dashlane і Sticky Password.

2.2. Методи вирішення задачі розробки програмного модуля захисту форм авторизації веб-сайтів

Для захисту даних користувачів, що користуються сучасними веб-сайтами, необхідно обмежити можливість повного перебору логінів та паролів, що зазвичай вводяться у форми авторизації. Для вирішення цієї задачі необхідно проаналізувати як відбувається перебор, розробити алгоритм захисту та реалізувати його у вигляді програмного коду.

Для наглядності потрібно створити універсальний приклад веб-сайту, що має можливість реєстрації користувача, форму авторизації (логін та пароль), поля з особистими даними користувача, для прикладу ім’я, прізвище, номер телефону, електронна адреса, країна. Для створення сайту необхідно скористатись технологіями HTML, CSS, JavaScript, PHP, SQL, що вже були розглянуті у першому розділі роботи.

На створеному веб-сайті треба провести дослідження, розробити алгоритм та код повного перебору паролів до логінів, аби імітувати атаку bruteforce. Створимо таблицю із даними, до яких логінів було підібрано пароль і за яку кількість спроб та час. Також ці дані будуть індикатором успішної атаки. Для неуспішної атаки також буде розроблено код, що буде виводити дані про потенційного порушника.

Задля економії часу та ресурсів у експерименті паролі будуть використані невеликих значень, тобто двох, трьох та чотирьохзначні числа. Для перебору більших значень потрібні потужні комп’ютери та багато часу, а для демонстрації захисту від атаки - складність паролю не має значення.

Після проведення експерименту необхідно буде проаналізувати його хід та результати для розробки алгоритму захисту, що потім буде реалізований у вигляді коду та імплементований на сайт. Після активації захисту проведемо ще один експеримент для демонстрації його ефективності. Буде створено користувачів із текстовими логінами та числовими паролями з 3 та 4 знаків. У ході атаки буде

проведено спроби підібрати паролі повним перебором чисел від 100 до 9999 використовуючи мову JavaScript та консоль веб-браузера. Необхідно розробити інтерфейс для відображення даних результатів експериментів у вигляді таблиць. Потрібно провести ескперименти до та після впровадження модулю захисту.

Результати перших експериментів повинні стати базою для подальшого дослідження методу підвищення захищеності форм авторизації веб-сайтів і на їх основі необхідно розробити програмний модуль захисту.

2.3. Розробка алгоритму та коду для імітації атаки повним перебором.

На основі обраного у пункті 2.2 методу, необхідно розробити алгоритм перебору паролів. Варіації можуть бути різні, для наглядності візьмемо ситуацію, при якій відомий логін користувача, а пароль невідомий. Для цього необхідно перебрати усі можливі варіанти паролів. Перебор може бути за словником, або повний перебор кожного можливо значення по зростанню. Для економії часу та ресурсів під час проведення експериментів, оберемо метод перебору чисельних паролів зі значенням від 100 до 9999. Якщо брати більші значення або використовувати окрім чисел букви та символи, експеримент може зайняти значно більше часу та потребувати більшої потужності системи, на якій він проводиться. Для проведення експерименту взлому та подальшому захисту від цього взлому в цьому немає потреби, дані з використанням паролів від 100 до 9999 дадуть репрезантативні результати, що будуть справедливими і для інших значень.

Блок-схема запропонованого алгоритму представлена на рисунку 2.1. Такий алгоритм зручно та ефективно реалізувати використовуючи мову JavaScript, оскільки така програма зможе бути запущена без додаткового ПЗ, це можна зробити прямо у консолі браузера, наприклад Google Chrome.

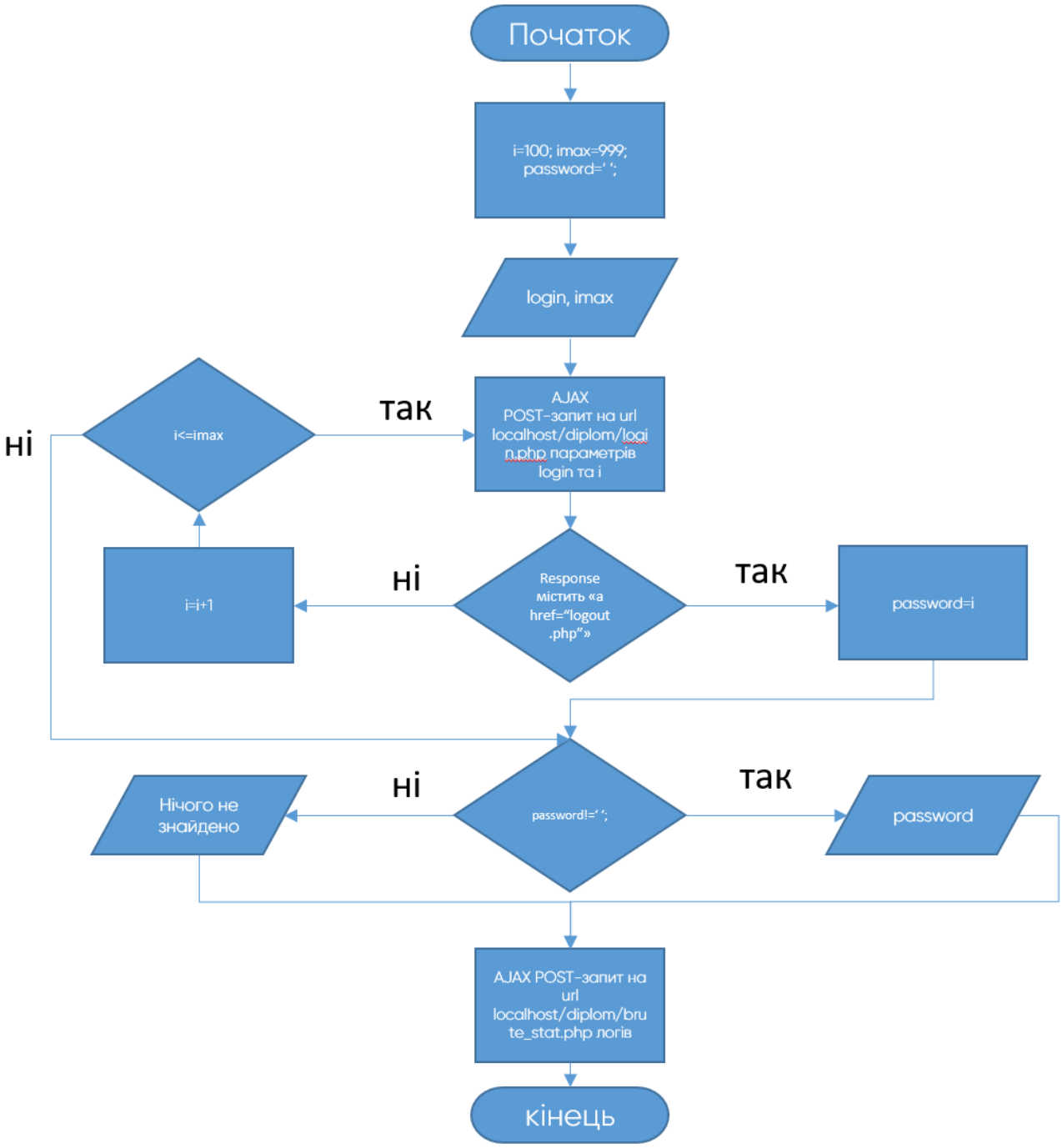


Рис. 2.1

На основі розробленого алгоритму створимо програмний код. Для імітації атаки повним перебором створений мовою JavaScript та застосовується через консоль веб-браузера. Цей код необхідно застосувати у консолі веб-бразуера, після чого викликати функцію brute(), в аргументах якої потрібно вказати у лапках бажаний логін, для якого відбувається підбір паролю та максимальне значення для перебору (необов’язково, за умовчанням до 9999). Вихідний код даної програми:

```
function brute(login, range = 9999) {
 let start = new Date().getTime();  let prevResult = '';
 let password = '';  let attempts = 0;
 for (let i = 100; i <= range; i++) {
  console.log(`Attempt #${++attempts}`);
  $.ajax({      type: 'POST',      url: '',      data: {"login": login, "password": i},
   success: function(data) {
    if (data.indexOf('href="logout.php"') !== -1) {
     password = i;        }      },
   async:false    });
  if (password !== '') break; }
 if (password !== '') {    console.log('Success!!!\n', `Login is: ${login}\n`,
`Password is: ${password}`);
 } else {    console.log(`Password not found`);  }
 let end = new Date().getTime();
 let time_ms = end - start;
 console.log(`SecondWay: ${time_ms}ms`);
 $.ajax({
  type: 'POST',
  url: 'brute_stat_log.php',
  data: {
   "login": login,
   "password": password,
   "attempts": attempts,
   "time_ms": time_ms,
  },  });
}
```

Для запуску процесу підбору необхідно у консолі викликати функцію brute(); (рис. 2.2) і в якості аргументів указати в лапках логін, до якого потрібно

підібрати пароль, і як другий аргумент вказати максимальне значення паролю (за замовчуванням 9999).

Рис. 2.2

Важливо використовувати складні та довгі паролі. На рисунку 2.3 представлена інфомрація щодо швидкості підбору різних паролів, якщо вважати швидкість перебору 100 млн. паролів на секунду. Те що прості паролі дуже швидко зламуються повинно стати важливим аргументом для власників веб-сайтів, аби вводилась перевірка на складність паролів. Потрібно вимагати від користувачів налаштовувати собі складні паролі, що мають в одній комбінації і великі, і малі літери, цифри та символи.

Набор символов	Длина пароля	Пример паролей	Время для полного перебора
A .. Z	5	CRUEL	мгновенно
A .. Z	6	SECRET	3с
A .. Z	7	MONSTER	1м 23с
A .. Z	8	COOLGIRL	36м 11с
A .. Z, 0 .. 9	5	COOL3	мгновенно
A .. Z, 0 .. 9	6	BANG13	22с
A .. Z, 0 .. 9	7	POKER00	13м 26с
A .. Z, 0 .. 9	8	LETMEBE4	8ч 3м 37с
A .. Z, a .. z, 0 .. 9	5	P0k3r	9с
A .. Z, a .. z, 0 .. 9	6	S3cr31	9м 37с
A .. Z, a .. z, 0 .. 9	7	DidIt13	9ч 56м 33с
A .. Z, a .. z, 0 .. 9	8	GoAway99	25д 16ч 26м 34с

Рис. 2.3

2.4. Висновки до другого розділу

В ході роботи над цим розділом було розроблено метод вирішення задачі захисту форм авторизації веб-сайтів, а саме проведення експериментальних атак задля отримання даних, що дозволять розробити ефективний алгоритм захисту. Розроблено алгоритм повного перебору та його програмний код мовою JavaScript, що дозволить провести практичний експеримент, на основі аналізу результатів якого можна буде сформувати алгоритм захисту форм авторизації веб-сайтів.

Розділ 3. РОЗРОБКА ТА ЕКСПЛУАТАЦІЯ ПРОГРАМНОГО МОДУЛЮ ЗАХИСТУ ФОРМ АВТОРИЗАЦІЇ ВЕБ-САЙТІВ

3.1. Налаштування локального серверу

WAMP - це скорочення від Windows, Apache, MySQL та PHP. Цей програмний стек, що означає, що встановлення WAMP встановлює Apache, MySQL та PHP у вашій операційній системі (Windows у випадку WAMP). Якщо ми не можемо встановити їх окремо, вони, як правило, з’являються в комплексі, і якщо є інші причини. Розберём, что такое WAMP.

WAMP відбувається від LAMP (L означає Linux). Єдине відмінність між ними полягає в тому, що WAMP використовується для Windows, а LAMP - для операційної системи на базі Linux.

«W» обозначає Windows, є також LAMP (для Linux) і MAMP (для Mac).

«A» означає Apache. Apache - це серверне програмне забезпечення, яке відповідає за обслуговування веб-сторінок. Коли ви запрашиваете сторінку, яку ви побачите, Apache задовольнить ваш запит через HTTP та запропонує вам веб-сайт.

«M» означає MySQL. MySQL - це система управління базами даних для вашого сервера. Вона зберігає всю необхідну інформацію, таку як вміст вашого веб-сайту, профілі користувачів та т. д.

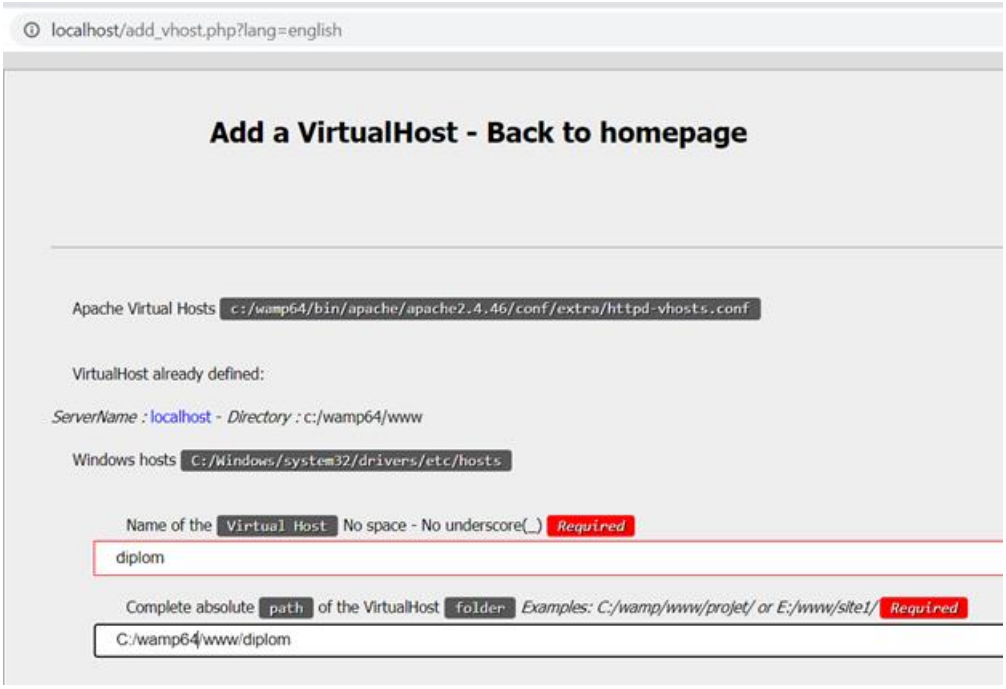
«P» обозначає PHP. Це мовна програма програмування, яка була використана для написання WordPress. Це діє як клей для всього цього програмного стека. PHP працює спільно з Apache та взаємодіє з MySQL.

Замість того, щоб встановити та перевірити WordPress на своїй навчальній сторінці хостингу, ви можете зробити це на своєму персональному комп'ютері (localhost).

Сервер WAMP діє як віртуальний сервер на вашому комп'ютері. Він дозволяє вам протестувати всі функції WordPress без будь-якого результату, оскільки він локалізований на вашому комп'ютері та не підключений до інтернету. Відповідно, WAMP використовується як безпечне місце для роботи на вашому сайті, без необхідності розміщувати його в мережі.

WAMP також має панель управління. Після встановлення пакетів програмного забезпечення всіх служб, що використовуються вище (для підключення операційної системи), будуть встановлені на вашому локальному комп'ютері.

Встановлюємо WAMP та переходимо за адресою localhost для створення віртуального хосту із назвою diplom (рис. 3.1).



(рис. 3.1)

Перейдемо до файлу hosts (рис. 3.2) та додамо рядки:

127.0.0.1 diplom

::1 diplom

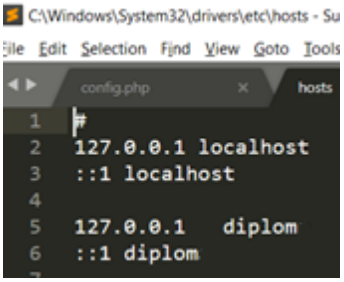


Рис. 3.2

Тепер наш сайт буде доступний за адресою “localhost/diplom”. WAMP дозволяє швидко обирати версію мови PHP, оберемо версію 7.3.21 (рис. 3.3)

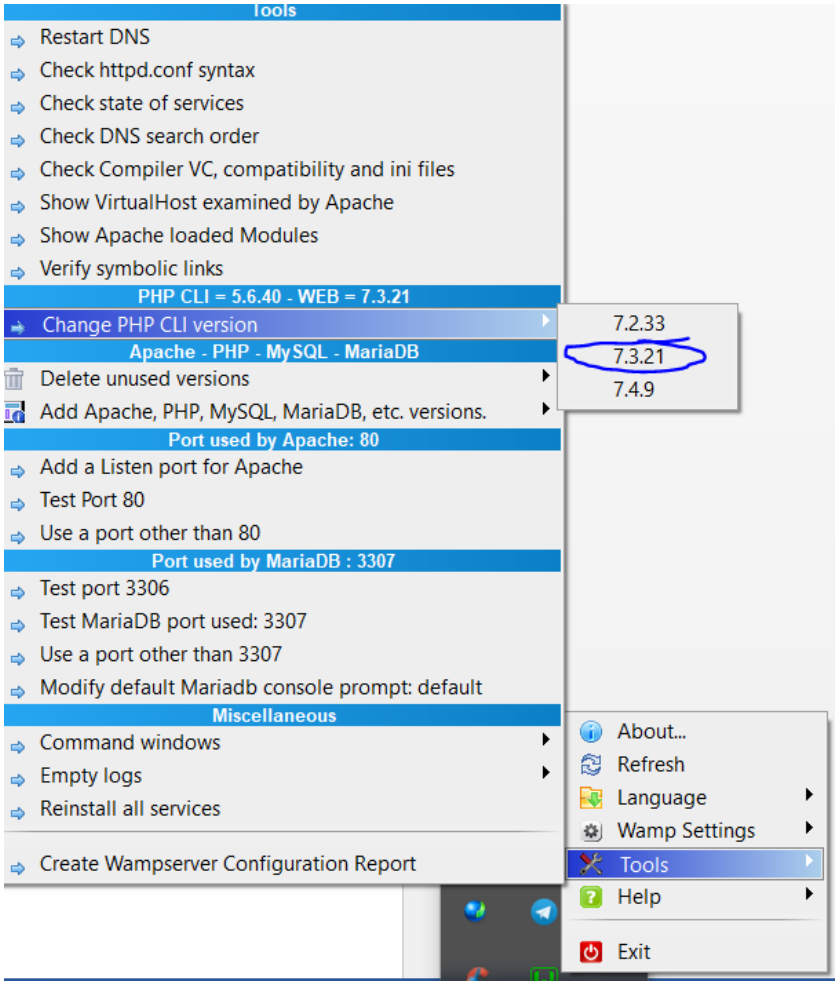


Рис. 3.3

Таким чином проведено базові налаштування локального серверу, що дозволяють встановити на нього розроблений приклад типового сучасного веб-сайту для проведення подальших експериментів із атаками повним перебором на форми авторизації та захистом від цих атак.

3.1. Розробка та налаштування веб-сайту для проведення експериментів

Опис структури сайту. Типовий сучасний сайт має основні споріднені елементи, що стосуються зокрема і даної роботи. Чи це інтернет-магазин, чи соціальна мережа, чи інтернет-банк – абсолютна більшість з великий компаній мають на свої сайтах схожі функції для користувачів, а саме реєстрацію та авторизацію.

Перебуваючи на сайті банку, користувач вирішує зайти в особистий кабінет, щоб зробити грошовий переказ. На сторінці особистого кабінету система спочатку просить ввести ідентифікатор. Це може бути логін, ім'я та прізвище, адресу електронної пошти або номер мобільного телефону.

Який конкретно вид даних необхідно ввести - залежить від ресурсу. Дані, які вказувалися при реєстрації, необхідно ввести для отримання доступу. Якщо при реєстрації вказувалося кілька типів даних - і логін, і адреса електронної пошти, і номер мобільного, то система сама підкаже що їй конкретно потрібно.

Введення цих даних необхідний для ідентифікації людини за монітором як користувача конкретно цього банку.

Якщо користувач в якості ідентифікатора ввів «Ілля Ільф», і система знайшла в своїй базі запис про користувача з таким ім'ям, то ідентифікація завершилась.

Після ідентифікації йде процес аутентифікації, в якому користувачеві потрібно довести, що він є людиною, яка реєструвався під ім'ям Ілля Ільф.

Для доказу необхідна наявність одного з типів аутентифікаційних даних:

- Щось, властиве тільки користувачеві. Біометричні дані: скани обличчя, відбитки пальців або сітківки ока.
- Щось, відоме тільки користувачеві. Сюди відносяться pin-коди, паролі, графічні ключі, секретні слова.

- Щось, що є в наявності у користувача. У даній якості може виступати токен, тобто компактний пристрій, призначений для забезпечення інформаційної безпеки користувача, також використовується для ідентифікації власника. Найпростіші токени не вимагають фізичного підключення до комп'ютера - у них є дисплей, де відображається число, яке користувач вводить в систему для здійснення входу; складніші підключаються до комп'ютерів за допомогою USB і Bluetooth-інтерфейсів.

Найпоширеніший тип аутентифікаційних даних - це пароль. Після введення користувачем пароля система перевіряє: чи відповідає умовний пароль «12345» користувачеві з ім'ям Ілля Ільф. Таким чином відбувається аутентифікація.

Якщо все вірно, і пара логін-пароль вірні, то система надасть користувачеві доступ до його ресурсів і здійснення банківських операцій, тобто відбудеться авторизація.

Описані процеси завжди відбуваються тільки в такому порядку: ідентифікація, аутентифікація, авторизація. Весь ланцюжок втратить сенс, якщо, наприклад, сайт спочатку надасть доступ до грошових коштів користувача, а потім буде уточнювати, чи це він насправді. Даний приклад справедливий не тільки для банків, а й для переважної більшості компаній, що мають сайт в інтернеті.

Отже, сайт повинен мати функції:

- реєстрації користувача (логін, пароль, особисті дані)
- авторизації, якій передують ідентифікація (введення логіну, що буде перевірятись на наявність у базі даних зареєстрованих користувачів) та аутентифікація (введення паролю та перевірка чи є він вірним до цього логіну)
- відображення успішної чи не успішної авторизації, тобто перехід до особистого кабінету користувача або текст про помилку авторизції

Для того, щоб можна було провести експериментальні дослідження, також будет створена сторінка із таблицями, що будуть містити дані експериментів, тобто таблиця про успішні спроби атак перебором, таблиця середньої швидкості взлому паролів різної довжини, таблиця порушників. Для усіх цих даних також потрібні таблиці у базі даних MySQL, що керується засобом phpMyAdmin.

Створення бази даних та необхідних таблиць. База даних - це впорядкований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. База даних зазвичай управляється системою управління базами даних (СКБД). Дані разом з СУБД, а також додатки, які з ними пов'язані, називаються системою баз даних, або, для стислості, просто базою даних.

Дані в найбільш поширених типах сучасних баз даних зазвичай формуються у вигляді рядків і стовпців в ряді таблиць, щоб забезпечити ефективність обробки і запитів даних. Потім можна легко отримувати доступ до даних, управляти ними, змінювати, оновлювати, контролювати та організовувати. У більшості баз даних для запису і запитів даних використовується мова структурованих запитів (SQL).

MySQL - це реляційна система управління базами даних з відкритим вихідним кодом на основі мови SQL. Вона була розроблена і оптимізована для веб-додатків і може працювати на будь-якій платформі. Так як з розвитком Інтернеті з'явилися нові вимоги, веб-розробники вважають за краще використовувати для веб-додатків платформу MySQL. База даних MySQL призначена для обробки мільйонів запитів і тисяч транзакцій, тому її часто вибирають компанії електронної комерції, яким потрібно керувати великою кількістю грошових переказів. Гнучкість у міру необхідності - основна характеристика MySQL.

Багато провідних веб-сайтів і веб-додатків використовують СУБД MySQL, в тому числі Airbnb, Uber, LinkedIn, Facebook, Twitter і YouTube.

Необхідно створити базу даних diplom_db (рис. 3.4), що буде мати таблицю користувачів (users), що міститиме такі дані, як логін, пароль (у подвійно

захешованому вигляді), ip адресу, ім'я, прізвище, телефон, країну, дату реєстрації на сайті. Також створимо таблицю (brute_stat). що необхідна буде нам для подальших експериментів.

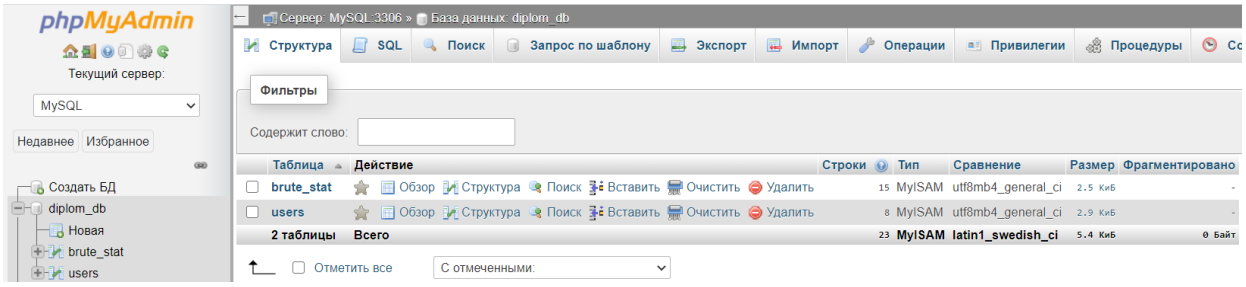


Рис. 3.4

Вигляд таблички users у інтерфейсі phpMyAdmin (рис. 3.5)

user_id	user_login	user_password	user_hash	user_ip	first_name	last_name	phone	country	reg_date
1	admin	d9b1d7db4cd6e70935368a1efb10e377	4c441d65d54c248002a3f1b630797843	0					NULL

Рис. 3.5

Значення user_hash генерується випадково і потім використовується для ідентифікації авторизованого користувача.

Пароль user_password це значення паролю, але подвійно захешоване алгоритмом MD5, що дозволить надійно захистити його, навіть якщо викрадуть це значення з бази даних, тому що на те, щоб його зламати – піде не один місяць.

Таблиця brute_stat та інші стовпці таблциі users буде розглянута детальніше у розділі присвяченому експериментальній атаці.

Розробка зовнішнього вигляду сайту засобами HTML, CSS. Розмітка створюється за допомогою HTML, стилі вигляду, шрифти, кольори за допомогою CSS, інтерактивні елементи (валідація) за допомогою JavaScript.

Код основної сторінки, де розміщений блок форми авторизації, який буде використаний потім для експериментів, наведено у додатку А.

Таблиця стилів, що була створена для покращення візуального сприйняття сайту та приближення його до сучасного дизайну, наведено у додатку Б.

Основна сторінка, код якої міститься у файлі index.php показана на рис. 3.6, при відкритті через браузер Google Chrome.

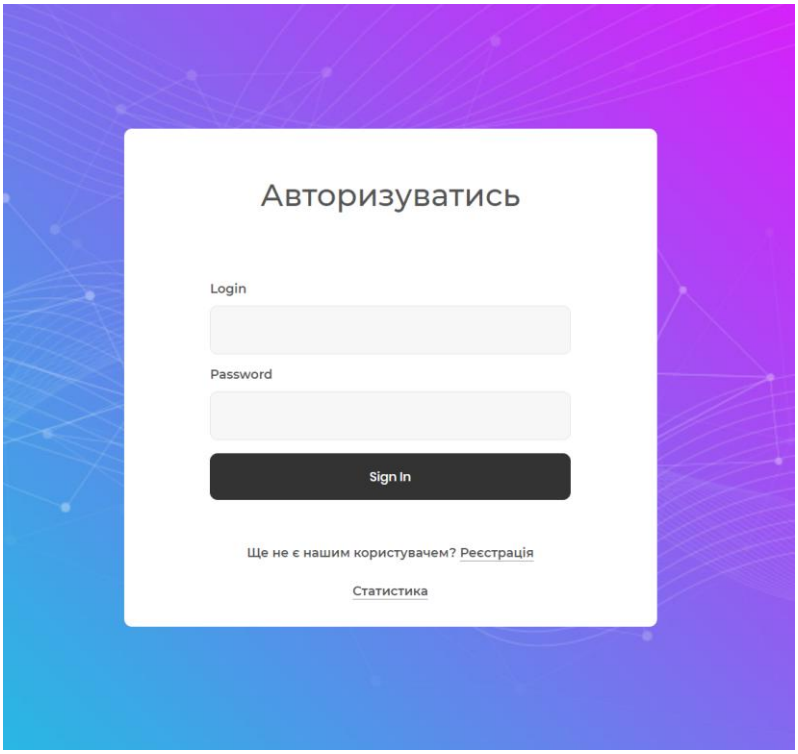


Рис. 3.6

Адреса головної сторінки на локальному сервері це localhost/diplom. Як можна побачити, на ній присутні обов'язкові елементи, що повинні бути присутні на сторінці де розміщена форма авторизації, а саме: поля для введення логіну, паролю, кнопка «Увійти» (англ. Sign Up), посилання на сторінку реєстрації, що розміщена за адресою localhost/diplom/registration.php. Посилання на сторінку «Статистика» знадобиться пізніше, аби мати доступ до сторінки де будуть показані дані експериментів.

Якщо користувач ще не зареєстрований, він має змогу перейти на сторінку реєстрації, файл registration.php, вихідний код якого надано у додатку В.

Сторінка реєстрація (рис. 3.7), що знаходиться за адресою localhost/diplom/registration.php, має наступні поля:

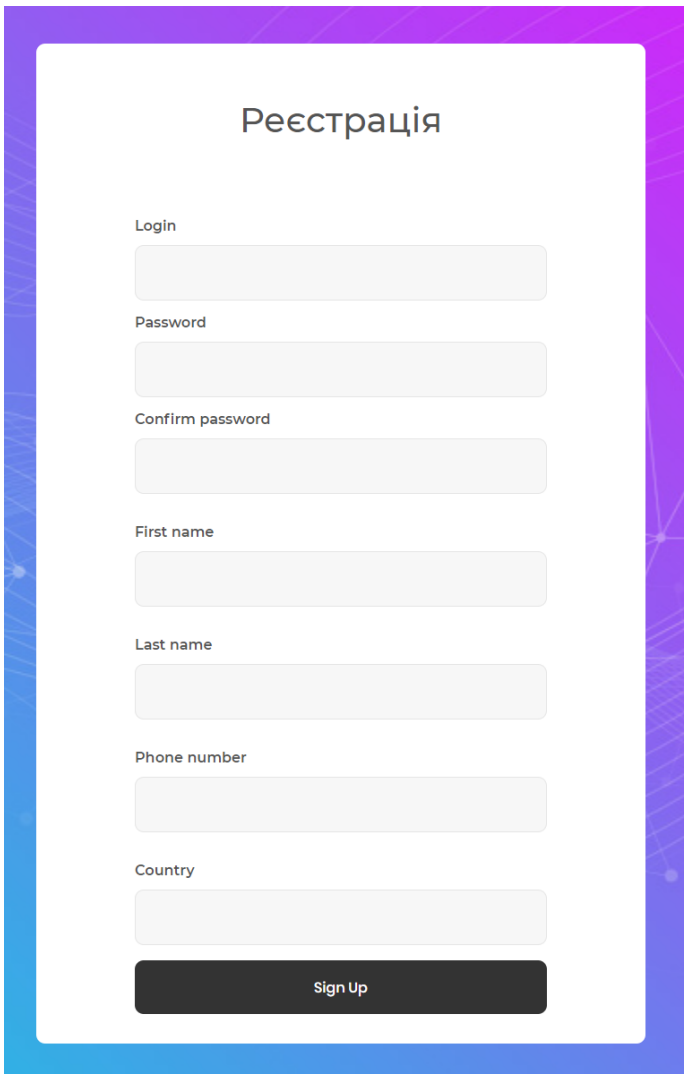


Рис. 3.7

- Login – ідентифікатор користувача
- Password – пароль
- Confirm Password – підтвердження (перевірка) паролю, аби користувач задав точно потрібний пароль під час реєстрації і не допустив помилку
- First name – ім’я
- Last name – прізвище
- Phone number – номер телефону
- Country – країна

Такі поля є стандартним прикладом форми реєстрації. В окремих випадках навіть ці дані можуть нести дуже велику цінність, що викликає потребу у захисті цих даних. До них можна отримати доступ, якщо підібрати логін та пароль до

форми авторизації, тому форма авторизації потребує додаткового захисту, який буде розроблено у подальшому ході роботи.

Для демонстрації особистого кабінету зареєструємо нового користувача. В якості паролю використаємо число 123 (рис. 3.8).

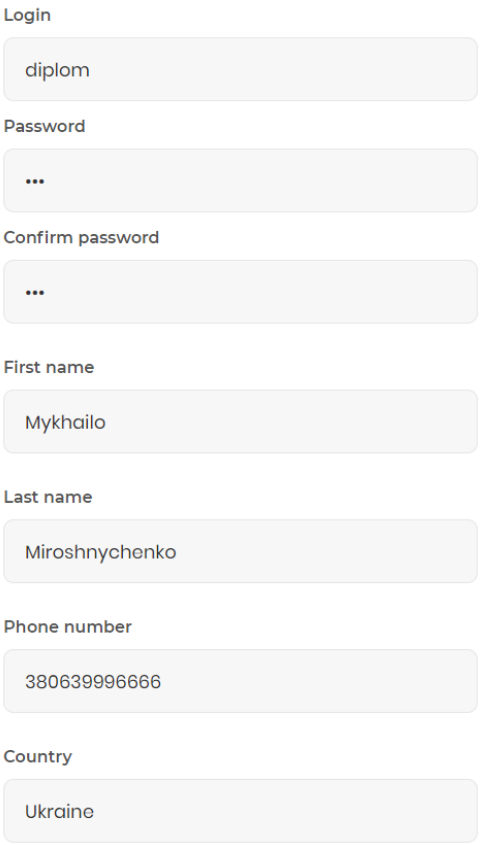


Рис. 3.8

Вихідний код особистого кабінету, в якому реалізовано відображення основних даних, що зазвичай зберігаються у база даних більшості сайтів, що надають послуги своїм клієнтам, надано нижче:

```
<?php include_once('app/account.php')?>
<?php include_once('common/header.php');?>
  <div class="limiter">
    <div class="container-login100" style="background-image:
url('images/bg-01.jpg');">
      <div class="wrap-login100 p-l-110 p-r-110 p-t-25 p-b-33">
        <div>
          <span class="login100-form-title p-b-30">
```

```
                                    Welcome to the account
                                </span>
                <br>
                <p style="font-size: 20px; font-weight: 500">
Your personal info:</p>
                <div class="p-b-10">
                    <div class="field-title">Login:</div>
                    <div class="field-val"><?=$userdata['user_login'];?></div>
                </div>
                <div class="p-b-10">
                    <div class="field-title">First name:</div>
                    <div class="field-val">
<?=$userdata['first_name'];?>
                    </div>
                </div>
                <div class="p-b-10">
                    <div class="field-title">Last name:</div>
                    <div class="field-val">
<?=$userdata['last_name'];?>
                    </div>
                </div>
                <div class="p-b-10">
                    <div class="field-title">Phone:</div>
```

Адреса сторінки (відповідно до назви файлу account.php) є localhost/diplom/account.php. Одразу після реєстрації користувач потрапляє на цю сторінку (рис. 3.9).

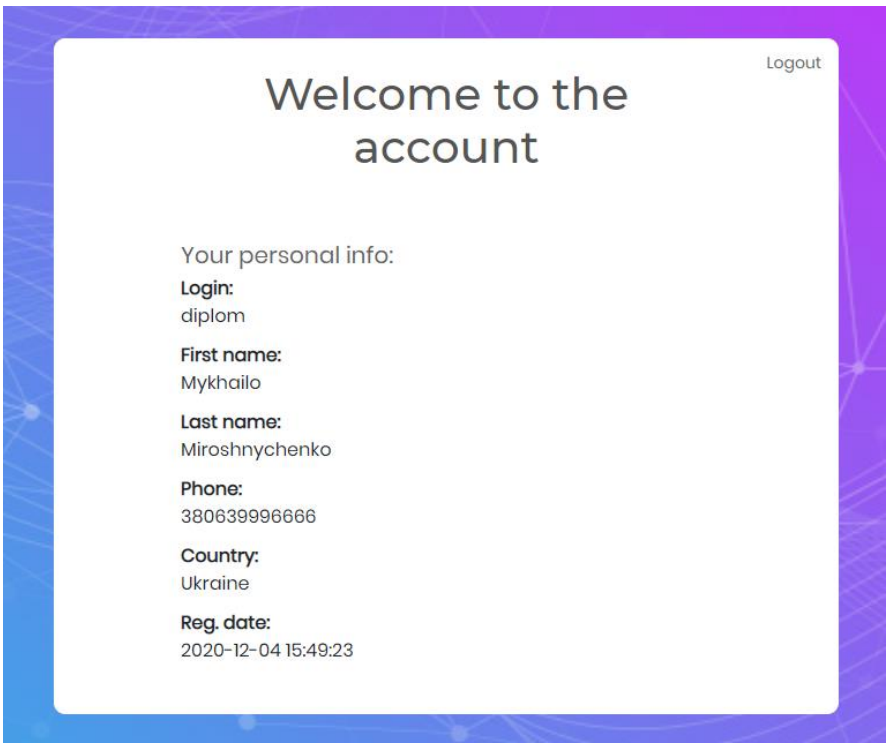


Рис. 3.9

На ній можна побачити повідомлення привітання, дані користувача, що були введені під час реєстрації та ще одне поле «Дата реєстрації» (англ. Reg. date). Також присутня кнопка «Logout» для виходу з особистого кабінету. У цей кабінет користувач також може потрапити при успішній авторизації з головної сторінки localhost/diplom.

Валідація необхідна для повідомлення користувачеві, що він некорректно заповнив поля тієї чи іншої форми.

Розробка коду валідації форм авторизації та реєстрації.

```
(function ($) {
    "use strict";
    var input = $('.validate-input .input100');
    $('.validate-form').on('submit',function(){
        var check = true;

        for(var i=0; i<input.length; i++) {
            if(validate(input[i]) == false){
                showValidate(input[i]);
```

```
                check=false;
            }
        }
        return check;
    });
    $('.validate-form .input100').each(function(){
        $(this).focus(function(){
           hideValidate(this);
        });
    });
    function validate (input) {
        if($(input).attr('type') == 'email' || $(input).attr('name') == 'email') {
            if($(input).val().trim().match(/^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.[0-
9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{1,5}|[0-9]{1,3})(\]?)$/) ==
null) {
                return false;
            }
        }
        else {
            if($(input).val().trim() == ''){
                return false;
            }
        }
    }
    function showValidate(input) {
        var thisAlert = $(input).parent();
        $(thisAlert).addClass('alert-validate');
    }
    function hideValidate(input) {
        var thisAlert = $(input).parent();
```

```
        $(thisAlert).removeClass('alert-validate');
    }
})(jQuery);
```

На головній сторінці код валідації генерує повідомлення про помилку, якщо користувач тисне на кнопку «Sign In» і при цьому не заповнив поля (рис. 3.10).

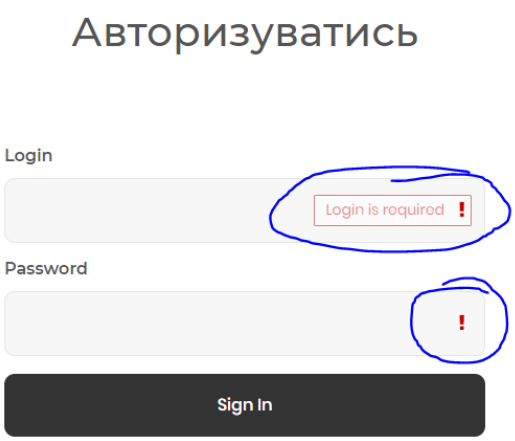


Рис 3.10

На сторінці реєстрації результат роботи скрипту ідентичний (рис. 3.11).

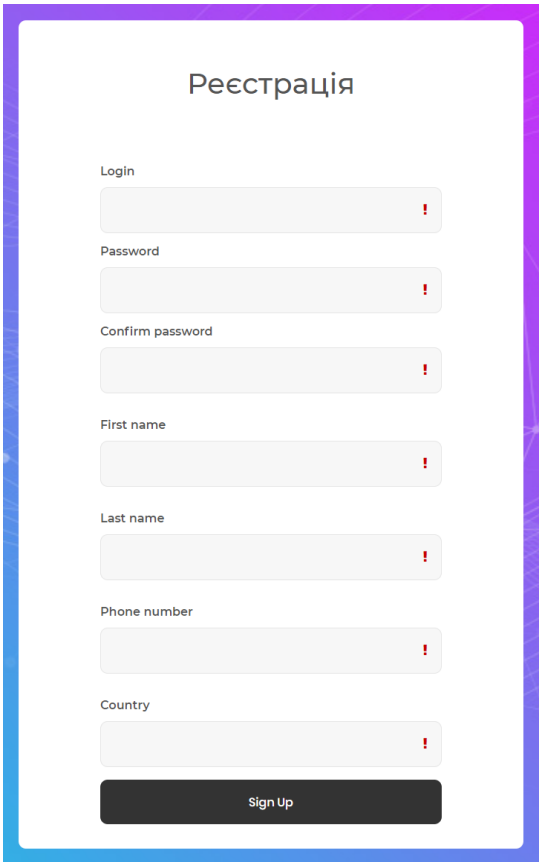


Рис. 3.11

Розробка backend функціоналу сайту за допомогою мови PHP. Спочатку потрібно створити зв’язок створеної раніше бази даних та коду нашого сайту. Необхідно задати значення аргументів функції mysqli_connect(), а саме ім’я хосту, користувача, пароль та назва бази даних. Для цього створимо файл config.php, де налаштуємо значення за допомогою коду:

```
<?php
define('DB_HOST', 'localhost');
define('DB_USER', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'diplom_db');
```

Для підключення створимо файл db.php із таким змістом:

```
<?php
include_once('config.php');
$db = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
```

Тепер при створенні якого-небудь php файлу, який потребує доступу до бази даних, достатньо на початку файлу використовувати код:

```
<?php
include_once('db.php');
```

Першим етапом взаємодії користувача та сайту, що має бути реалізовані мовою php – це реєстрація. Для цього створимо папку app у кореневому каталозі сайту (www). У ній створимо файл із назвою, відповідною до назви файлу, що містить код сторінки реєстрації. Створюємо файл app/registration.php

```
<?php
include_once('db.php');
if(!empty($_POST['login']))
{
   $err = [];
   if(!preg_match("/^[a-zA-Z0-9]+$/",$_POST['login']))
   {
```

```
            $err[] = "Логин может состоять только из букв английского алфавита
и цифр";    }
        if(strlen($_POST['login']) < 3 or strlen($_POST['login']) > 30)
        {
            $err[] = "Логин должен быть не меньше 3-х символов и не больше
30";    }
        if(strlen($_POST['password']) < 3 or strlen($_POST['password']) > 30)
        {
            $err[] = "Пароль должен быть не меньше 3-х символов и не больше
30";
        }
        if($_POST['password'] !== $_POST['password_confirm'])
        {
            $err[] = "Пароли должны совпадать";
        }
        $query = mysqli_query($db, "SELECT user_id FROM users WHERE
user_login='".mysqli_real_escape_string($db, $_POST['login'])."'");
        if(mysqli_num_rows($query) > 0)
        {
            $err[] = "Пользователь с таким логином уже существует в базе
данных";
        }
        if(count($err) == 0)
        {
            $login = $_POST['login'];
            $password = md5(md5(trim($_POST['password'])));
            $firstname = mysqli_real_escape_string($db, trim($_POST['firstname']));
            $lastname = mysqli_real_escape_string($db, trim($_POST['lastname']));
            $phone = mysqli_real_escape_string($db, trim($_POST['phone']));
            $country = mysqli_real_escape_string($db, trim($_POST['country']));
```

```
            $reg_date = date('Y-m-d H:i:s');
            mysqli_query($db,"INSERT INTO users
              SET user_login='".$login."',
              user_password='".$password."',
              first_name ='".$firstname."',
              last_name ='".$lastname."',
              phone ='".$phone."',
              country ='".$country."',
              reg_date ='".$reg_date."'
            ") or trigger_error("Query Failed! SQL:  - Error: ".mysqli_error($db),
E_USER_ERROR);
            include_once('login.php');
            header("Location: account.php"); exit();
          }
        }
      ?>
```

Цей код виконує такі функції:

- Перевірка логіну на коректність
- Перевірка паролю (confirm password) на введення та на коректність
- Перевірка, чи вже існує користувач з таким лоігном у базі даних
- Прибирання зайвих пробілів у введеному користувачем паролі та подвійне хешування паролю за допомогою алгоритму MD5
- В разі успішного проходження перевірок, створення запису у базі даних про створення нового користувача та передача туди його особистих даних

Якщо користувач вже зареєструвався, наступним етапом буде його авторизація. Для виконання цієї функції створимо файл app/login.php, вміст якого наведено у додатку Г. Розглянемо детальніше, що робить цей код:

- У ньому заданий код для генерації випадкового рядку, що буде використаний у хешуванні
- Пошук у базі даних логіну, відповідного тому, який вводить користувач при спробі авторизації (ідентифікація)

- Перевірка відповідності паролю введеного користувачем тому, який зберігається у базі даних (аутентифікація)
- Генерація хешу авторизації (згенероване випадкове число переведене за алгоритмом MD5) та внесення цих даних у базу даних у стовпчик user_hash
- Внесення даних про користувача (хеш та id) у Cookies
- Авторизація користувача до особистого кабінету

Таким чином мовою програмування PHP було розроблено функціонал притаманний більшості сучасних веб-сайтів, які мають особистий кабінет із даними користувача, форми реєстрації та авторизації, зв’язок із базою даних MySQL.

3.2. Експериментальне дослідження атаки повним перебором

Задля проведення експерименту нам знадобляться таблиці, де можна побачити результати експерименту. Створимо файл brute_stat.php (додаток Д). Він дасть змогу побачити успішні результати роботи розробленого раніше скрипту для атаки повним перебором. В одній з таблиць (рис. 3.12) ми зможемо побачити:

- Логін, пароль до якого був підібраний скриптом
- Підібраний вірний пароль
- Кількість спроб, яка знадобилась на підбір паролю
- Час, який зайняв підбір
- Час, коли була успішна спроба підібору вірного паролю

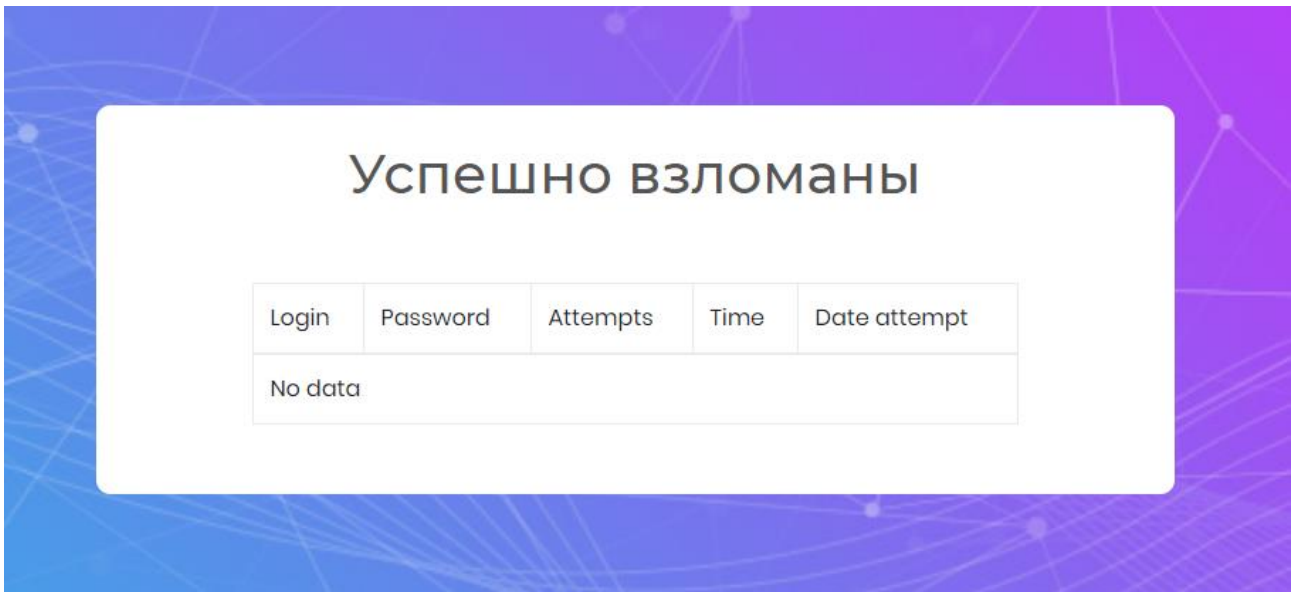


Рис. 3.12

Друга таблиця (рис 3.13) дасть змогу побачити надійність паролів різної довжини. У ній будуть виводитись довжина паролю (скільки знаків), кількість проведених експериментів (успішних спроб взлому паролю), середня кількість спроб підбору на один успішний експеримент, середній час проведення експерименту до досягнення успішного результату атаки.

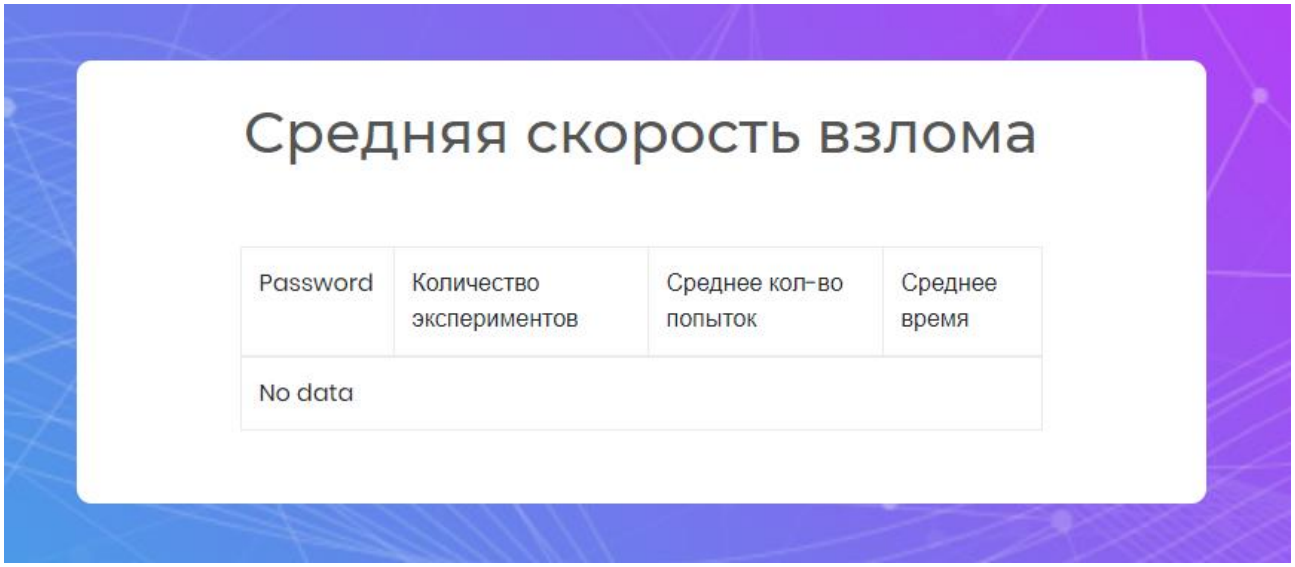


Рис. 3.13

Для проведення експерименту створимо користувачів із різними логінами та паролями, що наведені у таблиці 3.1.

Таблиця 3.1

Логіни та паролі користувачів для проведення експерименту

Логін	Пароль
test142	142
test777	777
test555	555
test1500	1500
test3200	3200
test8535	8535

Підтвердження створення цих користувачів бачимо у базі даних (рис. 3.14). Паролі представлені у вигляді хешу.

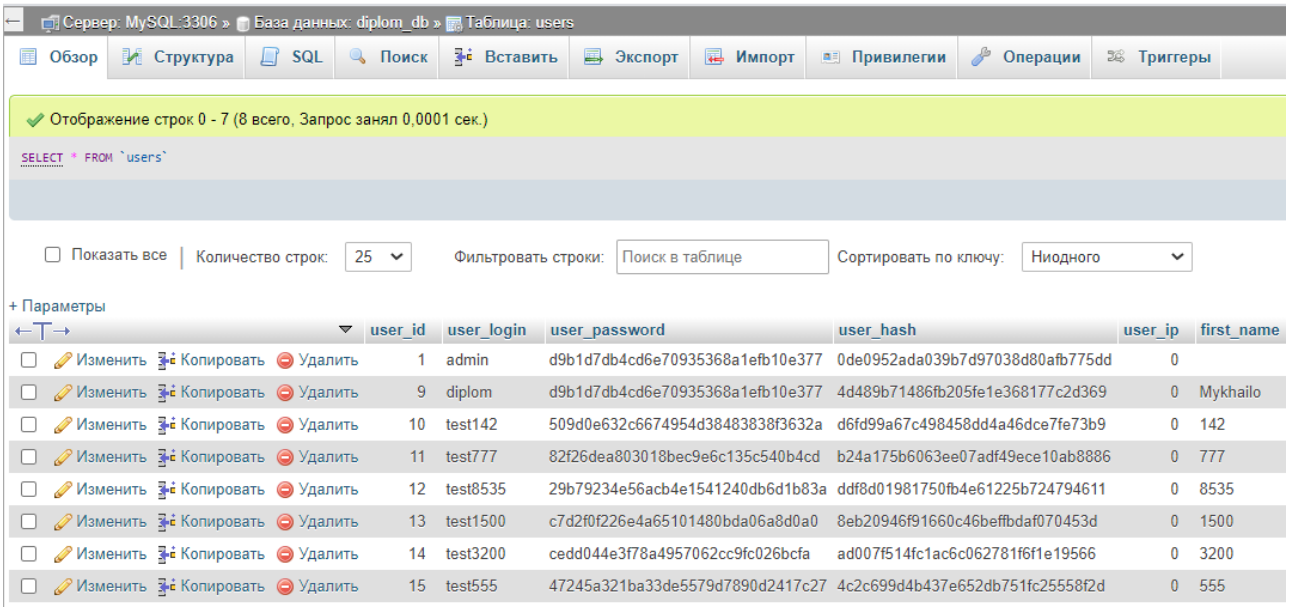


Рис. 3.14

Проведемо експериментальну атаку на кожен з цих логінів. Для цього застосуємо код наведений у додатку К в консолі браузеру Google Chrome, потім потрібно визивати функцію brute(), підставляючи в якості аргументу функції різні логіни. Уведемо даний код (рис. 3.15) та натиснемо Enter.

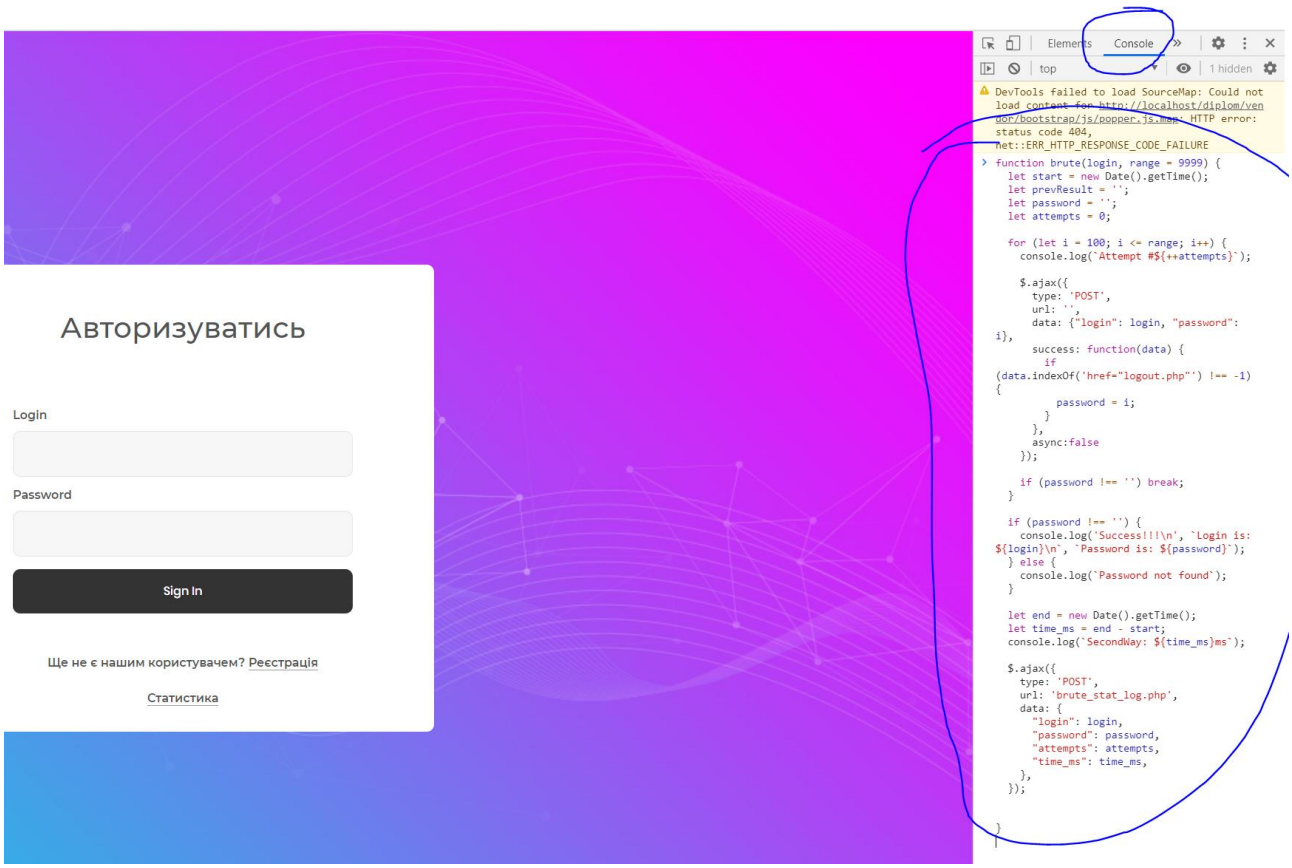


Рис. 3.15

У консолі викликаємо функцію brute(‘test142’); (рис. 3.16).

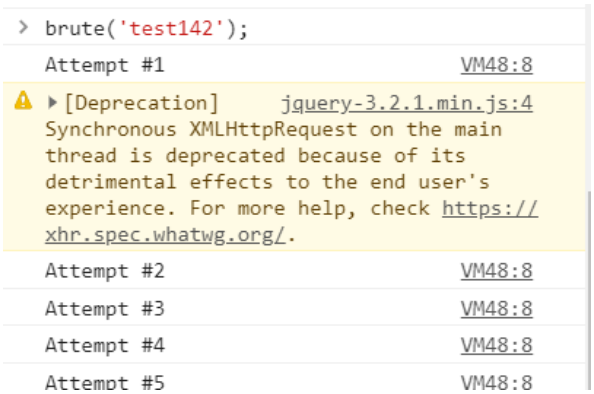


Рис. 3.16

Бачимо успішний результат виконання скрипту у консолі (рис. 3.17), пароль був зламаний на 43 спробі і ним виявилось значення «142».

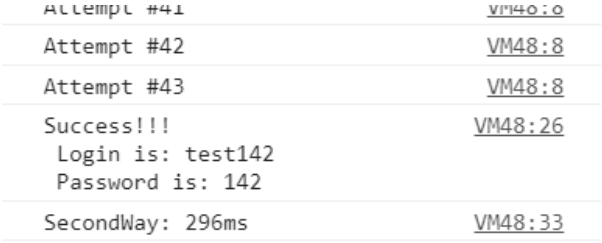


Рис. 3.17

Переглянемо першу таблицю на сторінці localhost/diplom/brute_stat.php і бачимо результат експерименту (рис. 3.18).

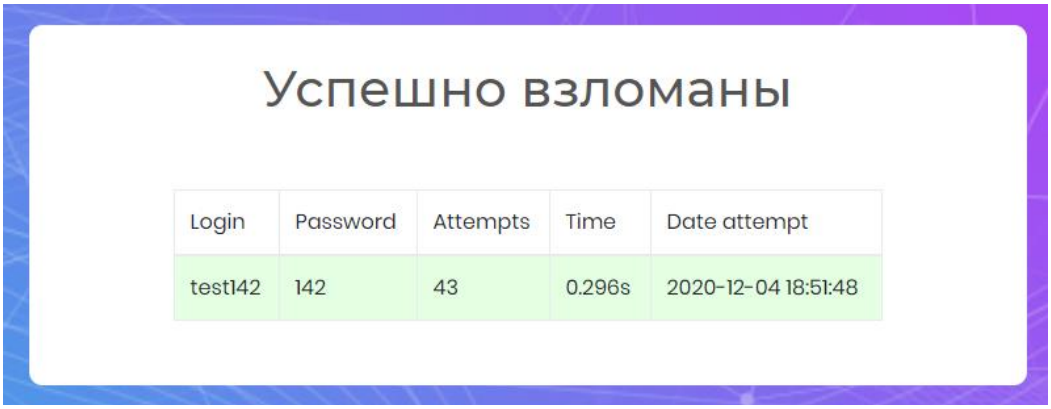


Рис. 3.18

Дані другої таблиці показані на рисунку 3.19.

Рис. 3.19

Проведемо аналогічний експеримент з іншими логінами, вказаними у таблиці 3.1. Бачимо результат на рисунках 3.20 та 3.21

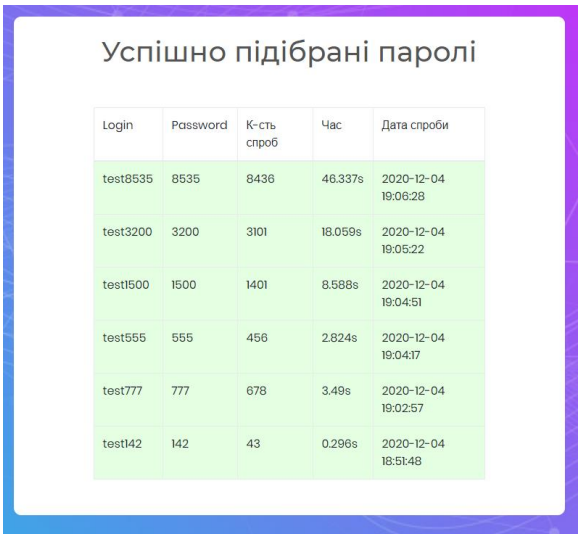


Рис. 3.20

Рис. 3.21

З результатів проведених досліджень можна зробити висновки:

- Код сторінки з формою авторизації не є надійним, якщо він не має додаткового програмного модулю захисту від атаки методом перебору. Навіть не складний скрипт може підібрати пароль, за умови достатнього часу та ресурсів.
- Безпека користувачів дуже сильно залежить від складності паролю. З другої таблиці сторінки localhost/diplom/brute_stat.php (рис. 3.21) можна побачити, що 3-х знаковий чисельний пароль зламується за лічені секунди, а саме середній час з трьох спроб є 2.203 секунди. Чисельний 4-х знаковий пароль був зламаний в середньому за 24.328 секунд.
- Атака зі словником може стати успішною за короткий термін, якщо прирівняти список найпопулярніших паролів із аналогічним перебором чисел. Якщо користувач має поширений пароль, він буде дуже швидко зламаний методом перебору варіантів найпоширеніших паролів, списки таких паролів є у вільному доступі в інтернеті.

На базі проведеного дослідження можна зробити такі рекомендації:

- Необхідно використовувати додатковий програмний модуль для захисту форм авторизації від атак методом повного перебору.

- Необхідно рекомендувати, або навіть змушувати користувачів використовувати складні паролі. Використання паролів, що мають велику довжину, можуть суттєво збільшити безпеку особистих даних користувачів. Паролі, що складаються тільки з чисел мають найменший спротив до атаки повного перебору, оскільки вони мають меншу кількість можливих символів порівняно з іншими наборами символів (табл. 3.2) і кількість спроб необхідних для їх підбору збільшується.
- Необхідно використовувати пароль, що не містить змістовних слів, найкраще використовувати генератори випадкових послідовностей, що задіюють усі можливі символи ASCII.

Таблиця 3.2

Кількість символів у різних наборах символів

Набір символів	Кількість символів
Арабські цифри	10
Шістнадцяткова система числення	16
Строчні букви латинського алфавіту	26
Усі печатні символи ASCII	95

Маючи ці результати експерименту, можемо перейти до реалізації захисту форм авторизації від атак перебором.

3.3. Розробка алгоритму захисту від атаки повним перебором

У проведеному експерименті в пункті 3.4 можна помітити особливість, що надасть змогу зробити ефективний захист форми авторизації від атаки повним перебором. Кількість спроб авторизації за короткий відрізок часу є надто великою для звичайної людини. Віслідкувавши кількість спроб на секунду, можна зробити висновок, чи реальна людина намагається авторизуватись, чи це

програма-бот виконує атаку перебором. На основі цієї особливості можна розробити алгоритм захисту. Він може бути невідчутним для користувача, навідміну від розповсюдженої технології CAPTCHA, яка потребує часу та уваги реального користувача і погіршує користувацький досвід, що може призвести до збитків компаніям та їх брендам.

Знаючи, що дуже велика кількість спроб на одиницю часу (наприклад на секунду) – це признак атаки, розробимо алгоритм, який буде блокувати можливість продовжувати спроби авторизації потенційно не реальним користувачам (тобто програмам-ботам), якщо кількість невдалих спроб авторизації за одним і тим самим логіном перевищує одну на 3 секунди, то повинно з’явитись попередження, аби наступна спроба була повторена через 3 секунди. У разі, якщо це людина, яка просто швидко намагалася авторизуватись, прочитавши це повідомлення вона зачекає і здійснить наступну спробу через 3 секунди. Якщо це повідомлення ігнорується і продовжуються численні спроби авторизації, то з великою імовірністю це атака, яку необхідно блокувати, тому додаємо ще 3 секунди часу за кожну наступну спробу авторизації, що відбулася у термін до закінчення попередньої затримки. Таким чином програма атаки не зможе обійти затримку часу і не зможе продовжити підбір паролів. Якщо відбудеться більше 20 спроб авторизації при активній затримці, то інформація про цього порушника буде виведена у третій табличці на сторінці localhost/diplom/brute_stat.php з інформацією про логін, час останньої спроби авторизації, ip адресу порушника. Блок-схема алгоритму показана на рисунку 3.22.

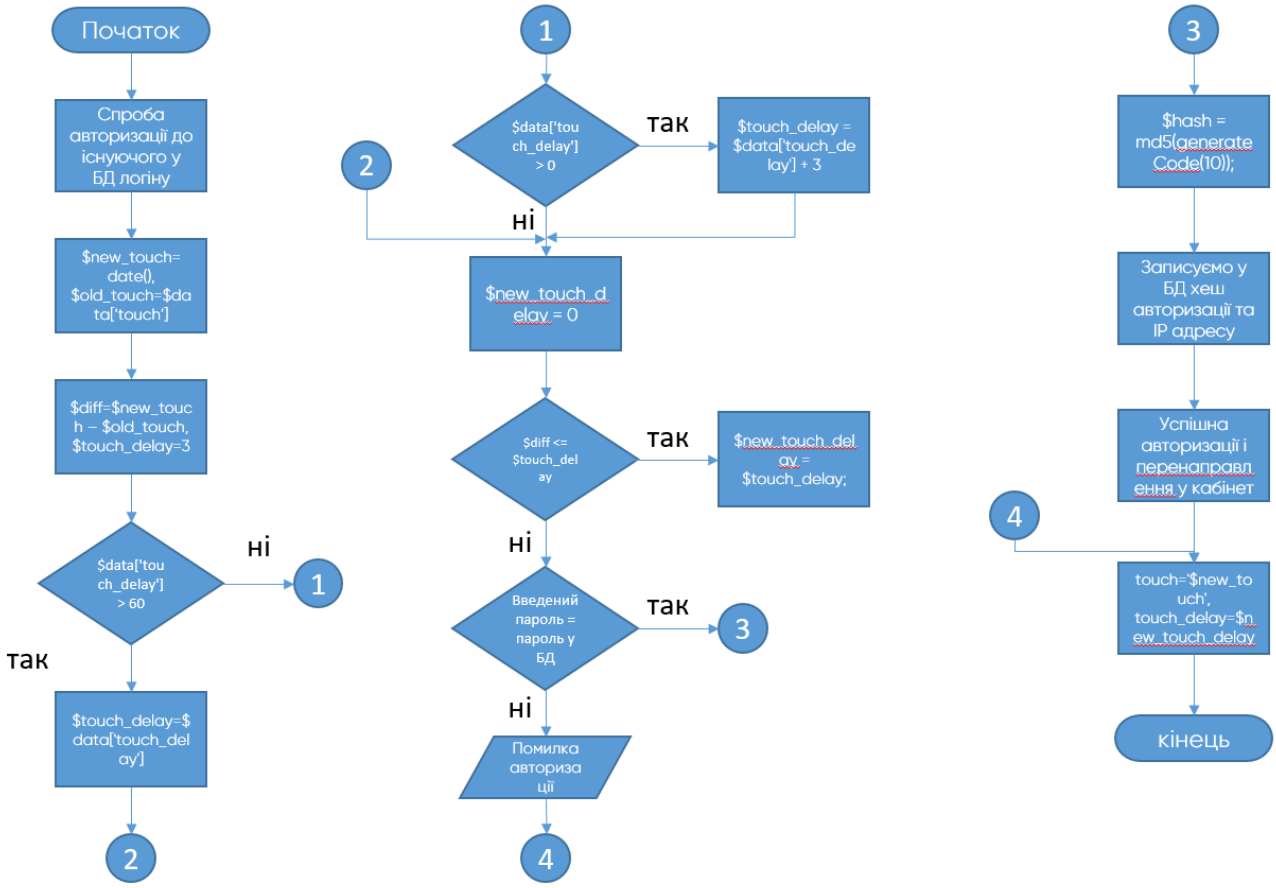


Рис. 3.22

3.4. Розробка програмного модулю захисту форм авторизації мовою PHP на основі розробленого алгоритму

Створимо код, що реалізує алгоритм захисту. Його необхідно інтегрувати у файл app/login.php, який відповідає за функції авторизації на головній сторінці.

```
if(isset($_POST['login']))
{
    $query = mysqli_query($db,"SELECT user_id, user_password, touch, touch_delay FROM users WHERE user_login='".mysqli_real_escape_string($db,$_POST['login'])."' LIMIT 1")
```

```
        or trigger_error("Query Failed! SQL:  - Error: ".mysqli_error($db),
E_USER_ERROR);
        $data = mysqli_fetch_assoc($query);
        if (!empty($data)) {
            $new_touch = date('Y-m-d H:i:s');
            $old_touch = $data['touch'];
            $diff = strtotime($new_touch) - strtotime($old_touch);
            $touch_delay = 3;
            if ($data['touch_delay'] > 60) {
                $touch_delay = $data['touch_delay'];
            } else if ($data['touch_delay'] > 0) {
                $touch_delay = $data['touch_delay'] + 3;
            }
            $new_touch_delay = 0;
            if (BRUTE_BLOCK && $diff <= $touch_delay) {
                $err[]= "Ошибка авторизации. Попробуйте через $touch_delay сек";
                $new_touch_delay = $touch_delay;
            } else {
                if($data['user_password'] === md5(md5($_POST['password']))) {
                    $hash = md5(generateCode(10));
                    $insip                              =                              ",
user_ip=INET_ATON('".$_SERVER['REMOTE_ADDR']."')";
                    $sql = "UPDATE users SET user_hash='".$hash."' WHERE
user_id='".$data['user_id']."'";
                    mysqli_query($db, $sql)
                    or trigger_error("Query Failed! SQL: $sql - Error:
".mysqli_error($db), E_USER_ERROR);
                    setcookie("id", $data['user_id'], time()+60*60*24*30, "/");
                    setcookie("hash", $hash, time()+60*60*24*30, "/", null, null, false);
                    header("Location: account.php"); exit();
```

```
        } else {
            $err[]= 'Ошибка авторизации';
        }
    }
    $ip_attempt = $_SERVER['REMOTE_ADDR'];
    $sql = "
        UPDATE users
        SET touch='$new_touch', touch_delay=$new_touch_delay,
ip_attempt='$ip_attempt'
        WHERE
user_login='".mysqli_real_escape_string($db,$_POST['login'])."'
    ";
    $query = mysqli_query($db, $sql)
    or trigger_error("Query Failed! SQL: - Error: ".mysqli_error($db),
E_USER_ERROR);
  }
}
```

У файлі config.php додамо рядок коду (рис. 3.23):

```
define('BRUTE_BLOCK', true);
```

де другий аргумент приймає значення true або false, і означає увімкнення або вимкнення модулю захисту.

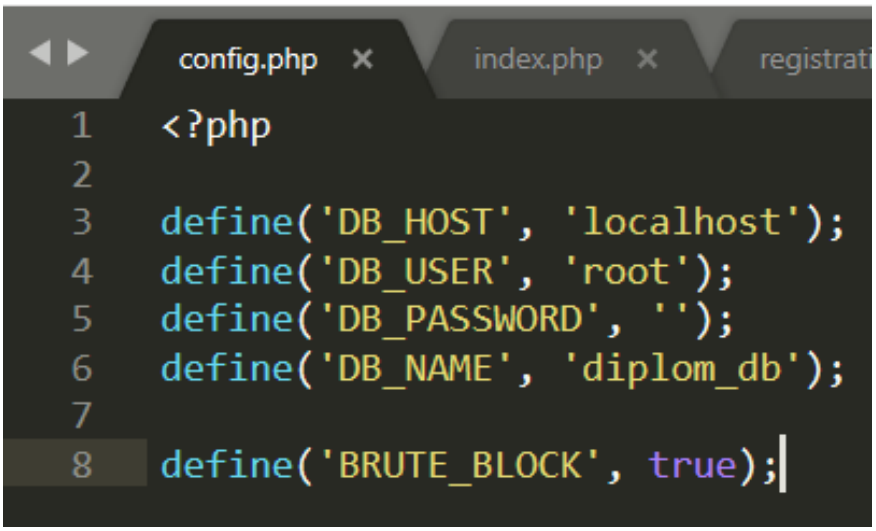


Рис. 3.23

Модуль захисту готовий і увімкнений, що дозволяє провести повторну експериментальну атаку і перевірити ефективність захисту.

3.5. Повторна експериментальна атака повним перебором та демонстрація роботи програмного модулю захисту форм авторизації

Проведемо повторний експеримент із використанням скрипту підбору паролів, що перебирає значення від 100 до 9999. Для цього повторюємо дії з минулого експерименту, викликаємо функцію brute(‘test142’);

На відміну від минулого ескперименту отримуємо зовсім інший результат. По черзі було проведено 9900 спроб для двох логінів і вдалого паролю не було знайдено для жодного з них, про що повідомляє текст Password not found у консолі браузера (рис. 3.24)

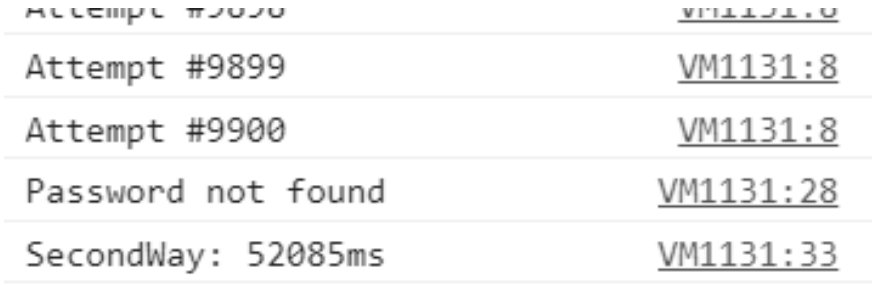


Рис. 3.24

Для відображення даних щодо порушників розробимо програмний код мовою програмування php та розмітки HTML, що дасть нам змогу бачити інформацію про відбиті атаки повним перебором одрауз на веб-сторінці.

```
<div class="limiter">
    <div class="container-login100" style="background-image: url('images/bg-01.jpg');">
        <div class="wrap-login100 p-l-110 p-r-110 p-t-25 p-b-33">
            <div>
                <span class="login100-form-title p-b-30">
```

```
                                    Відбиті атаки
                                </span>
                <br>
                <table class="table table-bordered">
                    <thead>
                    <tr>
                        <td>Login</td>
                        <td>Час останньої спроби</td>
                        <td>Ip</td>
                    </tr>
                    </thead>
                    <tbody>
                    <?php if (count($resultsFail)) { ?>
                        <?php foreach ($resultsFail as $resultFail) { ?>
                            <tr class="tr-red">
                                <td><?= $resultFail['user_login'] ;?></td>
                                <td><?= $resultFail['touch'] ;?></td>
                                <td><?= $resultFail['ip_attempt'] ;?></td>
                            </tr>
                        <?php } ?>
                    <?php } else { ?>
                        <tr><td colspan="100%">No data</td></tr>
                    <?php } ?>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
```

На сторінці localhost/diplom/brute_stat.php у третій табличці (рис. 3.25) бачимо дані про порушника: логін, до якого намагалися підібрати пароль, час останньої спроби, ip адресу.

Оскільки сайт налаштований на локальному сервері, виводиться адреса локального серверу «127.0.0.1» у скороченому вигляді «::1».

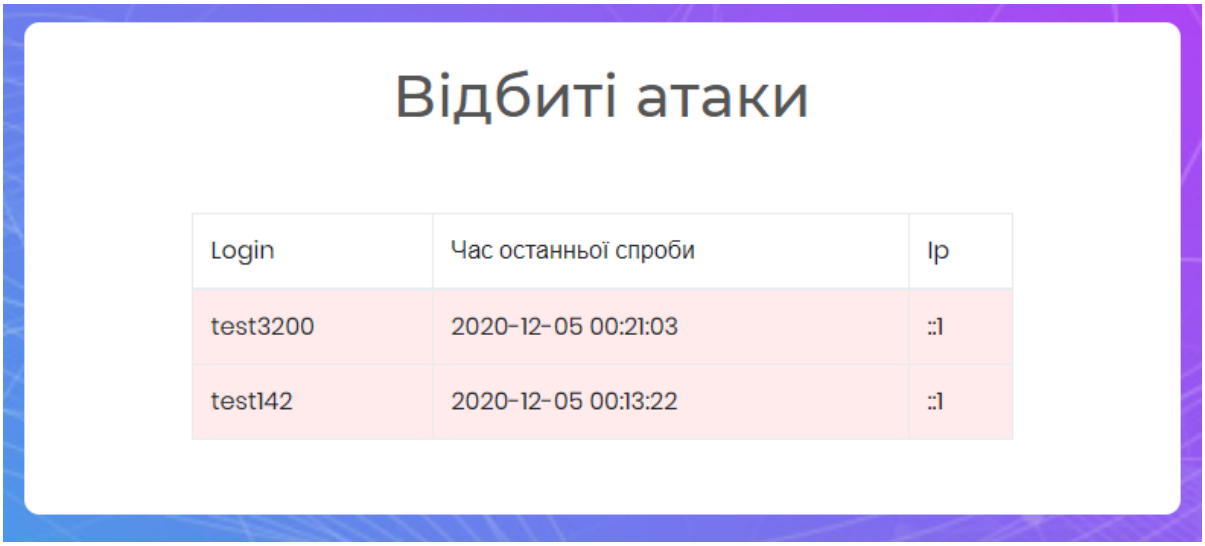


Рис. 3.25

3.6. Висновки до третього розділу

В ході виконання третього розділу магістерської роботи було отримано такі результати:

- Проведено експеримент (атака повним перебором за допомогою розробленого JavaScript коду), за рахунок чого було отримано дані, які знадобилися для подальшої розробки алгоритму захисту
- На основі аналізу результатів експерименту було створено алгоритм захисту, його блок-схему
- Створено програмний модуль захисту форм авторизації веб-сайтів від атак методом повного перебору

- Проведено експериментальне підтвердження ефективності програмного модулю захисту

Розроблений програмний модуль є ефективним методом забезпечення захисту форм авторизації від атак методом повного перебору, що дозволить компаніям гарантувати безпеку особистих даних користувачів їх веб-сайтів. Даний модуль розроблений мовою php і не потребує надто високої кваліфікації спеціаліста з безпеки або програмного інженера компанії для його встановлення та налаштування.

ВИСНОВКИ

Результатом виконаної роботи є програмний модуль захисту форм авторизації віб хакерських атак методом повного перебору. У процесі виконання роботи отримано наступні результати:

1. Проведено дослідження існуючих технологій створення веб-сайтів та їх вразливостей, виявлено, що технології створення та експлуатації веб-сайтів достатньо потужні і не потребують гострої потреби у подальших дослідженнях, але вразливості форм авторизації веб-сайтів становлять загрозу для користувачів.
2. Розроблено метод вирішення задачі підвищення захищеності форм авторизації веб-сайтів, а саме проведення експериментальних атак задля отримання даних, що дозволили розробити ефективний алгоритм захисту.
3. Проведено експериментальну атаку повним перебором, на базі результатів експерименту розроблено простий та ефективний алгоритм та програмний модуль захисту форм авторизації веб-сайтів мовою php, що дозволяє значно підвищити ефективність захисту особистих даних реальних користувачів

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Миллз Крис. Введение в HTML5 / Миллз Крис, Лоусон Брюс — М.: ИНТУИТ, 2015. — 134 с.
2. Клименко Р. А. Веб-мастеринг на 100%. — СПб.: Питер, 2013. — 512 с.
3. Макфарланд Д. Большая книга CSS3. 3-е изд. — СПб.: Питер, 2014. — 608 с.
4. Макфарланд Д. Новая большая книга CSS. — СПб.: Питер, 2016. — 720 с.
5. Кан М. Основы программирования на JavaScript — М.: Интуит, 2016. — 167 с.
6. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд. — СПб.: Питер, 2013. — 560 с.
7. Дэвис Е. М. Изучаем PHP и MySQL. / Дэвис Е. М., Филлипс Дж. А – Пер. с англ. – СПб: Символ-Плюс, 2008. – 448 с.
8. Рональд Р. Плю. Освой самостоятельно SQL за 24 часа / Рональд Р. Плю, Райан К. Стефенс – М.: Вильямс, 1998. – 352 с.
9. Что такое веб-сервер [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Learn/Что_такое_веб_сервер.
10. Padraic B. Книга «Как пережить полный конец обеда, или безопасность в PHP». Часть 1 [Електронний ресурс] / Brady Padraic. – 2016. – Режим доступу до ресурсу: https://habr.com/ru/company/mailru/blog/310726/.
11. Дружелюбная защита WEB ресурса от атак перебором [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://habr.com/ru/post/429804/.
12. Система CAPTCHA [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: https://www.bibliofond.ru/view.aspx?id=34021#_Toc136616302.
13. "Brute Force Attack", Imperva Glossary [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: http://www.imperva.com/application_defense_center/glossary/brute_force.html
14. iDefense: Brute-Force Exploitation of Web Application Session ID's ", By David Endler - iDEFENSE Labs [Електронний ресурс]. – 2006. – Режим

доступу до ресурсу: http://www.cgisecurity.com/lib/SessionIDs.pdf

15. "Protecting Secret Keys with Personal Entropy", By Carl Ellison, C. Hall, R. Milbert, and B. Schneier. [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: http://www.schneier.com/paper-personal-entropy.html
16. "Emergency Key Recovery without Third Parties", Carl Ellison. [Електронний ресурс]. – 1996. – Режим доступу до ресурсу: http://theworld.com/~cme/html/rump96.html
17. "Best Practices in Managing HTTP-Based Client Sessions", Gunter Ollmann - X-Force Security Assessment Services EMEA. [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: http://www.technicalinfo.net/papers/WebBasedSessionManagement.html
18. "Dos and Don'ts of Client Authentication on the Web", Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster - MIT Laboratory for Computer Science [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: http://cookies.lcs.mit.edu/pubs/webauth:tr.pdf
19. Cross Site Scripting Info. [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: http://httpd.apache.org/info/css-security/
20. "Analysis of format string bugs", By Andreas Thuemmel. [Електронний ресурс]. – 2004. – Режим доступу до ресурсу: http://downloads.securityfocus.com/library/format-bug-analysis.pdf
21. Reference "CERT © Advisory CA-2001-12 Superfluous Decoding Vulnerability in IIS" [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: http://www.cert.org/advisories/CA-2001-12.html
22. "Ravaged by Robots!", By Randal L. Schwartz [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: http://www.webtechniques.com/archives/2001/12/perl/

23. Карпова Т. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.
24. Мейер Д. Теория реляционных баз данных. – М.: Мир, 1987. – 608 с.
25. Гарсиа-Молина Гектор, Ульман Джеффри Д., Уидом Дженифер. Системы баз данных: Полный курс/Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1088 с.
26. Ульман Дж., Уидом Дж. Введение в системы баз данных/Пер. с англ. – М.: Издательство «Лори», 2000. – 374 с.