

Database Assignment 1

by Sameeran Raveenthiran

Submission date: 05-Dec-2022 11:39AM (UTC+0000)

Submission ID: 191042347

File name: 33405_Sameeran_Raveenthiran_Database_Assignment_1_460452_658925711.docx (552.92K)

Word count: 1274

Character count: 7243



DATABASES AND APPLICATION DEVELOPMENT



Sameeran Raveenthiran
Database-Driven Website Application

Contents

Two security issues that would need consideration if this website were to become a live website are:.....	2
SQL Injection Issue:	2
b) Identification and Authentication Failures:.....	2
Screenshot and description:	3
Make passwords secure (and update them regularly)	4
Automate website backups	4
Limit user permissions Establish a schedule for creating	5
Secure online checkouts	5
The second threat is the identification and authentication failure	5
Reference:	6

Two security issues that would need consideration if this website were to become a live website are:

SQL Injection Issue:

One of these potential issues is SQL Injection. Due to the fact, SQL injection consists of various methods such as malicious inserted into browsers or other forms of entry [3].

SQL injection is an issue to a website because attackers' identity can be hidden, and play with the existing data, cause repudiation problems, including changing balances or voiding transactions, enable full disclosure of all data including in the system, destroys data or makes the data unavailable and they can become the administrators of the database servers [3].

To have a safeguard of injection attacks, a certain method namely input validation method should be prioritised to make sure formatted data can be inputted, as a result leads to blocking malicious code from entering into the system [3].

Following these safety tips and practises can this website from SQL injections:

1. The best option to use a safe API, which abstains from using the interpreter fully, gives a parameterized interface, or relocates to Object Relational Mapping Tools (ORMs) [3].
2. Utilises positive server-side input validation. This isn't a complete version of defence due to various application require special characters, including text areas or APIs for mobile applications [3].
3. For any residual dynamic queries, escape special characters utilising the specific escape syntax towards that interpreter [3].
4. Use LIMIT and various other SQL controls that are within the queries to obviate mass disclosure of records in case of SQL injection [3].

b) Identification and Authentication Failures:

Another noticeable this website would be identification and authentication failures. Due to the fact, these are identified as vulnerabilities associated to applications authentication schemes. These failures can cause severe and damaging data breaches. To solve this issue of identification and authentication, is to make sure users and application are secure by utilising frequent web application penetration testing.

Some of the preventive measure from authentication and identification could be:

1. Create multi-factor authentication to avoid automated credential stuffing, stolen credential reuse attacks, and brute force [2].
2. Must not deploy or ship using any default credentials, especially for the admin users [2].
3. Create a weak password check, by testing new or changed passwords in the face of the worst 10.000 lists of passwords [2].
4. Arrange passwords length, rotation, and complexity with 'National Institute of Standards and Technology (NIST)) 800-63b's guidelines in section 5.1.1' towards Memorised Secret or other modern, evidence-based password policies [2].

5. Make sure registration, API pathways, credential recovery are hardened against account enumeration attacks, by utilising the identical messages for each outcome [2].
6. Delay or limit the failed login attempts warnings but be vary of creating a denial of service scenario. Log all failures and alert the administrators when attacks such as, brute force or credential stuffing are detected [2].
7. Utilise server side, built-in session manager that can produce new random session ID including high entropy after the login. Session identifier should be included in the URL, ensure you are securely stored, and invalidated after either idle, timeouts, or logout [2].

A. I have implemented defences against SQL injection in the given code:

Screenshot and description:

```
225
226 <!-- Modal -->
227 <div class="modal fade" id="loginmodal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
228   <div class="modal-dialog" role="document">
229     <div class="modal-content">
230       <div class="modal-header">
231         <h5 class="modal-title" id="exampleModalLabel">Login to Open Book Store</h5>
232         <button type="button" class="close" data-dismiss="modal" aria-label="Close">
233           <span aria-hidden="true">&times;</span>
234         </button>
235       </div>
236       <div class="modal-body">
237         <form method="post" action="./php/users.php">
238           <div class="form-group">
239             <label for="email">Email</label>
240             <input id="email" class="form-control" type="Email" name="email" required>
241           </div>
242           <div class="form-group">
243             <label for="password">Password</label>
244             <input id="password" class="form-control" type="password" name="password" required>
245           </div>
246         </div>
247         <div class="modal-footer">
248           <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
249           <button type="submit" class="btn btn-danger" name="login">Login</button>
250         </div>
251       </form>
252     </div>
253   </div>
254 </div>
255
```

```

254 </div>
255
256 <!-- login modal ends -->
257
258 <!-- Signup modal -->
259 <div class="modal fade" id="signupmodal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
260   <div class="modal-dialog" role="document">
261     <div class="modal-content">
262       <div class="modal-header">
263         <h5 class="modal-title" id="exampleModalLabel">New user? Please signup</h5>
264         <button type="button" class="close" data-dismiss="modal" aria-label="Close">
265           <span aria-hidden="true">&times;</span></button>
266       </div>
267       <div class="modal-body">
268         <form method="post" action="/php/users.php">
269           <div class="form-group">
270             <label for="name">Name</label>
271             <input id="name" class="form-control" type="text" name="name" required>
272           </div>
273           <div class="form-group">
274             <label for="email">Email</label>
275             <input id="email" class="form-control" type="email" name="email" required>
276           </div>
277           <div class="form-group">
278             <label for="password">Password</label>
279             <input id="password" class="form-control" type="password" name="password" required>
280           </div>
281           <div class="form-group">
282             <label for="password">Confirm Password</label>
283             <input id="cpassword" class="form-control" type="password" onkeyup="confirm()" name="cpassword" required>
284             <small class="form-text" id="helptxt"></small>
285           </div>
286         </div>
287       </div>
288       <div class="modal-footer">
289         <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
290         <button type="submit" id="signupbtn" name="signup" class="btn btn-danger" disabled>Signup</button>
291       </div>
292     </div>
293   </div>
294 </div>
295 <!-- Signup modal ends -->
296
297 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
298 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
299 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
300 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
301 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
302 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
303 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
304 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
305 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
306 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
307 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
308 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
309 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
310 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J864eJ10abPp7y1ZSLh0wvWFSQeV7S79sP08iF7wJ4w4HwAG70QZt3AJKAGn" crossorigin="anonymous"></script>
311 </script>
312 </body>
313 </html>

```

This is however a partial implementation of defence the important section of this website that needs to be improved in terms of security is HTTPS (Hyper Text Transfer Protocol Secure) which means using an SSL (Secure Socket Layer) to the information transmitted back and forth can't be seen by a third party. Other sections of this website and practices of security for preventing illegal access and fake authorization of this website, that need to improve are:

Make passwords secure (and update them regularly)

Most hacking related breaches which are around 80%, comes from passwords. A company can prevent that by implementing a security that can create unique and strong login credentials. You can use a password generator, namely LastPass to create a password that is very secure for your website. With the use of LastPass users can save and store the generated passwords. In addition, you can put a reminder in the calendar to frequently update the password [1].

Automate website backups

Implement a schedule for the purpose either creating website backups or automate the process through your website hosting a provider such as GoDaddy. Tools such as Easy cPanel Backup, eBackupper, or WordPress plugins can be utilized. like Vaultpress and UpdraftPlus can save and backup your website [1].

With a backup of your site, you can respond to a range of issues fast, whether it's a broken page or a hacked website. The problem, however, is that many businesses fail to back up their sites on a regular schedule. It's easy to solve this issue, though. Automate your website backups [1].

By having a backup, this can enable the user to react to issues much faster, whether the website has a broken page, or it has been hacked. The issue most businesses face is to backup their website by not following a schedule [1].

Limit user permissions Establish a schedule for creating

Giving everyone the ability to gain access and update your website can create a security vulnerability. Particularly If they fail to keep up with other items on the website's security checklist, such as updating passwords regularly. Which is why to increase security there must be a limit of user permissions.

Secure online checkouts

Online checkout security can be upgraded with the use of address verification system (AVS), additionally, credit card verification value (CVV) form fields for every checkout with credit card. Trusted AVS providers consists of Logate and Melissa. Utilizing a platform such as Shopify can aid you to create CVV fields automatically [4].

With an AVS, companies can equip an extra level of security towards online checkout process. The purpose of these system is to prevent fraudulent payments, which can result to loss of revenue and additional headache [4].

Other methods include Update plugins, CMS, invest in anti-malware software, get a DDoS mitigation service, and Preventing SQL injection attacks, which allow hackers to steal user data, by taking several proactive steps, including limiting user permissions, implementing data storage procedures, using whitelist input validation, and more. There are other methods such as CMS, update plugins, purchase anti-malware software, get a DDoS mitigation service, which will prevent SQL injection attacks [4].

The second threat is the identification and authentication failure

The second threat is the identification and authentication failure, and it can tremendously impact this website if we do not implement proper security preventive techniques and methods to safe and secure our website because Identification and authentication failures are vulnerabilities related to applications' authentication schemes. Such failures can lead to serious and damaging data breaches. Before deploying this website and launching live on the web it is imperative that I apply regular web application penetration testing, where possible, implement multi-factor authentication to prevent automated credential stuffing as well as using a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session identifier should not be in the URL, be securely stored, and invalidated after logout, idle, and absolute timeouts.

```

255
256 <!-- Login modal ends -->
257
258 <!-- Signup modal -->
259 <div class="modal fade" id="signupmodal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
260 <div class="modal-dialog" role="document">
261 <div class="modal-content">
262 <div class="modal-header">
263 <h5 class="modal-title" id="exampleModalLabel">New user? Please signup</h5>
264 <button type="button" class="close" data-dismiss="modal" aria-label="Close">
265 <span aria-hidden="true">&times;</span>
266 </button>
267 </div>
268 <div class="modal-body">
269 <form method="post" action="/php/users.php">
270 <div class="form-group">
271 <label for="name">Name</label>
272 <input id="name" class="form-control" type="text" name="name" required>
273 </div>
274 <div class="form-group">
275 <label for="email">Email</label>
276 <input id="email" class="form-control" type="email" name="email" required>
277 </div>
278 <div class="form-group">
279 <label for="password">Password</label>
280 <input id="password" class="form-control" type="password" name="password" required>
281 </div>
282 <div class="form-group">
283 <label for="password">Confirm Password</label>
284 <input id="cpassword" class="form-control" type="password" onkeyup="confirmpw()" name="cpassword" required>
285 <small class="form-text" id="helptxt"></small>
286 </div>
287
288
289 </div>
290 <div class="modal-footer">
291 <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
292 <button type="submit" id="signupbtn" name="signup" class="btn btn-danger" disabled=Signup</button>
293 </form>
294 </div>
295 </div>
296 </div>
297 <!-- Signup modal ends -->
298
299 <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6q4849b1E2+poT449yKv5vZP3S+P08lEj6bKJ71w6FAV0hwj1yVf0R53o2en" crossorigin="anonymous"></script>
300 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6C999G6pMvG8Q87F91V87I87Y87n3zV9z1TtL3UksdQ6V0dWf0oA6" crossorigin="anonymous"></script>
301 <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.4.1/dist/js/bootstrap.min.js" integrity="sha384-wf55QOpgSzdGog6h6GfqTb3UuibheggT8w1kX3Vs8mLX/qcdWs9z50gsBu0" crossorigin="anonymous"></script>
302 <script>
303 const queryString = window.location.search;
304 const urlParams = new URLSearchParams(queryString);
305 const msg = urlParams.get('msg');
306 if(msg !=undefined){
307 alert(msg);
308 }
309
310
311 </script>
312 </body>
313 </html>

```

Reference:

1. <https://www.webfx.com/blog/marketing/site-security-checklist/>
2. <https://www.hhs.gov/sites/default/files/owasp-top-10.pdf>
3. <https://snyk.io/learn/application-security/web-application-security/>
4. <https://cyolo.io/blog/identification-and-authentication-failures-and-how-to-prevent-them/>

Database Assignment 1

GRADEMARK REPORT

FINAL GRADE

61 / 100

GENERAL COMMENTS

Instructor

WEBSITE

Database structure and design - Structure is OK - arguably you might want to consider normalising the author to another table 9/10

Range of PHP demonstrated -

Can only search by book title I believe

Validation is OK, but could be a bit more extensive. Login is generally well managed, but I was able to access favourite.php via the URL. This should be locked off if not logged in.

It's a bit strange to be asked for name etc when borrowing a book - I'm already logged in so the store knows who I am already.

Although there is edit and delete in the code, I cannot find any way in the website to use it.

28/50

Overall integration of website - /10

Once logged in, it's not immediately obvious what extra features there are, a landing page after login would be nice. Generally a good layout and navigation however. 7/10

REPORT

Investigation/analysis of two security issues -SQL

Injection is generally good, I would have liked a few more examples other than the 1=1 problem that was covered in class. You have also not mentioned prepared statements which we have covered in class several times. Identification and authentication failure is good 9 /15

Mitigation of security issues in your site - You need to use prepared statements. Your examples don't show the database query to be injected?? I think you may have misunderstood this. Proposals for authentication failure avoidance a reasonable and suitable. 8/15

General Comments: Quite strange that you have written PHP in a very different style to that studied in class. Don't forget to reference if you're going to do this.

All academic reports should be structured with an introduction and conclusion - a short one would be OK here given the low word count

TOTAL: 61/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4
