

yr2 sem1 - db web driven app report (1702201)

by Jericho Jermaine Uzzell

FILE	12014_JERICO_JERMAINE_UZZELL_YR2_SEM1_- _DB_WEB_DRIVEN_APP_REPORT__1702201__294270_260221362.DOCX (49.56K)		
TIME SUBMITTED	18-JAN-2021 12:38PM (UTC+0000)	WORD COUNT	1886
SUBMISSION ID	142530039	CHARACTER COUNT	10029

University of Wales Trinity Saint David

School of Computing

Database-driven website

ACCA5022

Jericho Uzzell

1702201

10/01/2021



Prifysgol Cymru

Y Drindod Dewi Sant

University of Wales

Trinity Saint David

1. Introduction; [page 3]
2. Site pages overview; [page 3-4]
 - 2.1. Home; [page 3]
 - 2.2. Community page; [page 3]
 - 2.2.1. Posts;
 - 2.2.2. Friends;
 - 2.2.3. Users;
 - 2.3. Registering and logging in; [page 3-4]
 - 2.3.1. Registering; [page 3]
 - 2.3.2. Logging in; [page 4]
 - 2.4. Settings page; [page 4]
3. Security; [page 5-6]
 - 3.1. What is security? [page 5]
 - 3.2. Cross site scripting; [page 5]
 - 3.3. Cross site request forgery; [page 6]
 - 3.4. SQL Injections; [page 6]
 - 3.5. General security; [page 6]
4. Ready for deployment? [page 7]
5. Conclusion; [page 7]
6. Reference; [page 7]

1. Introduction

The site is another social media! Its intention is to allow users to post comments, likes, and posts. You can search for friends, register, upload images and more. The site is built on some simple rules, these being: the user can control their privacy for profile, posts, and both; we value privacy and security.

This document will go over the pages within this site, the security we believe is relevant and whether this site is ready for deployment – then concluding this with a verdict.

2. Site pages overview

2.1. Home page

The home page has two different types of view. One for logged in and one for logged out users – there is a requirement for different versions as logged in users should not be able to see outstanding friend requests, recent friends posts etc.

Both versions (logged in and out) offer the ability to show recent public posts.

2.2. Community page

2.2.1. Posts

Having a dedicated page for posts is ideal as it can allow for sharing posts around the community and to others who are not registered. Posts can inherit privacy that the users setting is currently on, for example, if their privacy is set to anyone can view, the post will be available to anyone.

There is a feature where you can manually define the privacy for each post from anyone can see, only friends can see and lastly no one but me can see. A feature the student intends to add is, everyone except the following people can view my post and only the following people can view my post.

2.2.2. Friends

The friends page under the community tab, shows your current friends, sent friend requests and received friend requests. This page offers the ability to remove a friend, revoke a friend request, accept and deny a friend request. The student intends to have an ability to block users and private message users.

2.2.3. Users

The page shows the users page, this will include their posts, comments, likes and activity if their privacy allows it. This page should link direct messages, images the user has uploaded and more.

The student intends to have social media links such as YouTube, Twitter and more.

2.3. Registering and logging in

2.3.1. Registering

Registering with Wiz is simple, it requires a username, firstname, surname, email and password. Once all fields are valid and the user clicks sign up, the system will email the user with an authentication email to verify the account. Passwords must follow certain rules, such as minimum length, numbers and/special character and at least one capital letter.

2.3.2. Logging in

Logging in requires a username and password, all login attempts are logged. After so many failed attempts, the account is locked and can only be reactivated via resetting the password. All failed attempts are informed to the user once they are logged in, they can report any suspected malicious attempts.

2.4. Settings page

The settings page bundles some key categories together, these being icons/profile picture, privacy setting and friends. This page gives you information on when the user joined Wiz and will offer an ability for the user to edit their first and last name.

3. Security

3.1. What is security?

Security for the web and other entities should always be considered by the end user, the designers or applications and the organisation. How does the end user evaluate the security of the website they are currently on?

Reviewing the website's privacy policy and terms and conditions can give you an understanding, it gives the end user an understanding of what information is stored and who is it shared with.

Another clue on the website's security could be as simple as resetting your password. If the website sends you back your password when you click forgot password, this is an extreme concern which means you should avoid this website.

Security can cover other areas, according to [5] the types of vulnerabilities are: Cross site scripting (XSS), SQL Injection (SQI), denial-of-service (DoS), distributed denial-of-service (DDoS), Memory corruption, Buffer overflow, Cross-site-request-forgery (CSRF), Data breach; we will discuss the more common which will be below.

3.2. Cross site scripting (XSS)

Cross site script, also known as XSS occurs when an attacker or unintended user, manipulates a vulnerable website, where it will send back JavaScript inside a victim's browser – the attack can compromise the users interaction with the application according [1].

What does that mean? Essentially, if you type into a box on a website and enter within that box, valid HTML (hyper-text mark-up language). If that text you entered returns those valid HTML tags and is executed as code, then that can be manipulated. If the site does not validate anything to do with the text, and simply returns it, it can be a security floor. The types of tags sent could be "" for bold (essentially without closing it, it can make the entire site bold) or worse, "<script>".

If the text returned is only visible to you (this could be note on a site for only you), then this is not a problem as such, as you are the only person that will see it. With that said, if that text is not sanitised on that note as an example, then you can bet other parts of that site to be the same. However, if the text box is public facing, and the data is available to other end users, this becomes a problem.

According to [1], there are three types of XSS. Reflected XSS comes from the current HTTP, example of this could be if you typed in the URL (Uniform Resource Locator) valid html code.

[1] Gives an example of "https://insecure-website.com/status?message=<script>/*+Bad+stuff+here...+*/</script>".

Stored XSS is when text is stored on the sites database, an example of this could be when you look at someone's post.

Lastly, DOM-based XSS is a vulnerability on the client-side rather than server-side, an example of this would be with Google and if you searched "Google" it would return everything as bold (Google sanitised the text in reality)

3.3. Cross site request forgery (CSRF)

Cross site request forgery, also known as CSRF is an attack where the end user executes unwanted actions on a website they have logged into, according to [2]. Actions can include but are not limited to: transferring funds, changing emails or phone numbers, messaging other users and more; [2] continues to state that if the user account is an administrator, this can compromise the entire website.

How exactly can an attacker do this? According to [2], social engineering such as sending emails or chat messages can do this. Some basic examples [2] provides are URLs the attacker could provide, such as GET requests "http://bank.com/transfer.do?acct=BOB&amount=100" and POST requests which could have a hidden value within a form on a separate website or the same website which is compromised with XSS.

How can a website avoid being the victim of CSRF? According to [3], a token for POST data could be used. They state that CSRF tokens should strong and unpredictable, they suggest you should use cryptographic strength pseudo-random number generator (PRNG), seeded with timestamps. [3] Goes on to state how these tokens should be transmitted, as a hidden input field within HTML code, however the student also believes have the token generated within JavaScript (JS) is also fine.

[3] States to valid these codes, the generated stored one (within the user's session data) should be compared with the POST/GET data. If it matches, this is valid, if there is no token sent or it does not match, then you can deny the request.

3.4. SQL Injections

An SQL Injection or SQLI attack (we will refer to it as SQLI), is an attack on a website where code is injected where hackers can maliciously bypass systems to gain access to the SQL database according to [4]. This is done when the end user is adding, editing or other types of data where they can extract information from databases, tables, and entities within them. [4] Continues to state that the hackers look to delete the database/table/entities or even to steal data such as passwords.

How do you protect against SQLI? [4] States you should blacklist known characters that can cause these issues, you should use prepared statements and parameterised queries. They further state that you should ensure your database is on the latest updated version.

4. Ready for deployment?

The security features introduced within this application includes all of the above, being CSRF, XSS and SQLi protection. The application sanitises data returned from the database that had user interaction to prevent XSS and CSRF. The post data sent to the server-side code all requires a token unless the code is for logged out users. The tokens are generated every so often and are also quite long and unpredictable.

The server-sided code uses PHP PDO prepared statements with parameterised queries.

With all of the above stated, posts are counted sequentially instead of a random number -- have a sequential number is not bad however this should only be for administration and backend coding. For public facing entities, you should avoid sequential counting as users can just think, if this post is 5, what is post 4 and what is post 6?

Although privacy policies do apply, it does not fit well for competitors and small communities within the site. Competitors can figure how many posts you have on the site and small communities could be bombarded with spam. This without considering bots reading the page and grabbing names. Instead it should be a randomly generated string with numbers, this would be a simple fix.

With that security floor regarding posts outlined, the site still has undeveloped features, desired features and perfections to come. The student stated that this site, although somewhat done, is nowhere near their expectations and they would not deem it ready for deployment -- the features required to be developed are home pages features, community page features and settings page.

5. Conclusion

Concluding this document, the student needs to complete the website before deployment can be considered. Once the website has been completed, then it can be deployed. The posts security issue can be allowed to continue however, it must be amended at the earliest opportunity.

6. References

- [1] PortSwigger, "What is cross-site script (XSS)", *PortSwigger*, 10 January 2021, [Online], Available: <https://portswigger.net/web-security/cross-site-scripting>, [10/01/2021]
- [2] S.Kirsten, "Cross Site Request Forgery (CSRF)", *owasp*, 10 January 2021, [Online], Available: <https://owasp.org/www-community/attacks/csrf>, [10/01/2021]
- [3] PortSwigger, "CSRF tokens", 10 January 2021, [Online], Available: <https://portswigger.net/web-security/csrf/tokens>, [10/01/2021]
- [4] insightsforprofessionals, "What is SQL Injections & Howto protect yourself", *insightsforprofessionals*, [Online], Available: https://www.insightsforprofessionals.com/it/security/5-minute-guide-to-sql-injection?gclid=Cj0KCQiA6Or_BRC_ARIsAPzuer_hKFN0ITrwmtCX_c8nCOGLmYcJZz5O3OBvdYWmvUlq9-aKBh_LJbMaApyoEALw_wcB, [10/01.2021]
- [5] cloudflare, "Web Application Security Learning Objectives", cloudflare, [Online], Available: <https://www.cloudflare.com/en-gb/learning/security/what-is-web-application-security/>, [10/01/2021]

yr2 sem1 - db web driven app report (1702201)

GRADEMARK REPORT

FINAL GRADE

88/100

GENERAL COMMENTS

Instructor

WEBSITE

Database structure and design - it's not clear if all of the tables are in use as some don't have rows in them, but nevertheless, fully normalised and highly complex 15/15 -

Range of PHP code demonstrated - One or two small Notices showing - you could suppress this as mentioned in your video (or turn off notices in the server) Log in is well managed. Good validation on the friends especially that you can't add yourself as a friend. You could make the button invisible on your own profile so the user isn't confused. I don't think it was actually possible to edit text, although, you could change the image and items such as privacy etc, so you have demonstrated the UPDATE skill. Search is shown on users - there could be some more extensive searching for the purpose of demonstrating your skills. Good use of validation. Overall you have demonstrated a very good grasp of the technology and moved your knowledge beyond that covered in class. Although you state that the site is unfinished, there are easily enough features for the scope of this assignment. Well done. 37/40

Overall integration of website - There are some gremlins in the structure which wouldn't allow it to port cleanly to my server, but thanks for the video demo which I requested which clearly showed the website working. Some pages are a little blank, particularly the early pages, you might want to put a little more content to give the user some signposting through your site. However overall, the navigation and layout is fine. 13/15

REPORT

Security issues identified 11/15 - You have identified some of the main features that affect websites such as these. You might also want to talk about SSL deployment to encrypt data - especially as your website deals with personal details.

Consideration of security issues for website - I would have liked a little more detail (eg some code extracts) to support your statement that you have implemented against XSS/CSRF/SQL Injection. I was able to find it in your code though! Some good work here. 12/15

A very good assignment overall, well done. 88%

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7
