Security

Security is one of the most important tasks in Information Technology. The tasks of keeping the information of a company, a website, data, and policies secure, is extremely important. Security vulnerabilities are hard to detect, but with the right plans in place, information can be protected from prying eyes. I plan to discover several ways to secure our project, using secure usernames and passwords. If passwords and data are encrypted, they can be hard for non-authorized users to detect. The task of security will also entail testing, setting up firewalls, and helping in other areas. Supplying the data to be entered in the database is one of the areas I will be discussing here.

Our team name is Bug Overflow, and our database will describe the bugs found in and outside of your home. Say for instance you found a bug in your basement that looks weird and you have no idea of what kind of bug it is, where it came from, and why it is there. You can take a picture of the bug, upload it to our site, and it will give you the name and information of that bug. This is a process and we are bound to run into some problems along the way. At that point we must swarm them, mobilizing the person best fit to solve it. The goal to swarming is to contain the problem before it can spread. It is necessary to prevent other problems from progressing downstream. This can affect cost and effort to repair it, and can increase technical debt. Swarming enables learning, and that is why testing is so important also.

In creating the admin usernames and passwords for each team member, the site I used to get secure usernames is Jimpix username generator. The usernames that were generated from the site, were changed around and different characters were added. Some letters were changed to be sure they are unique. Then the passwords were created from Norton’s identity safe site. The same technique was used to change the usernames, to change the passwords to make them unique also. The next step is to set up an admin email account using Google. Source code integrity and code signing should be used meaning all developers should have their own PGP keys created in a managed system, and for this project we will use GitHub.

The data to be stored into the database has been created, and pushed to GitHub. There are thousands of bugs in the world and we’ve come up with a few unique ones, and some common bugs that you may see every day. We want to keep our data secure from SQL injections, and this will require manual and automated testing. Tests that can run continuously in our deployment pipeline. One such tool that can be used is Gauntlt, which puts its security code in Gherkin syntax scripts. Used by a wide variety of developers for unit and functional security testing on every committed change such as static code analysis. We can do this by deploying a static analysis. A Static analysis tool will inspect program code for possible run-time behaviors and seek out coding flaws, errors and backdoors that could be potentially malicious code (known as testing from the inside-out). An example of this is Brakeman, Code Climate.

We could also use Dynamic analysis tool which consists of testing while the program is running, which can monitor items in the systems memory, functional behavior, and response time, (known as testing from the outside-in). It would be wise to also include dependency scanning performed at build time, inside of a deployment environment. This involves inventorying all our dependencies for binaries and executables, making sure they are free from malicious binaries and vulnerabilities. Examples are Maven for Java, and the OWASP (Open Web Application Security Project) Dependency-Check (Kim, pg. 667.9/880).

Applications must be secure by using firewalls, and using tools such as Nmap to ensure that only expected ports are open, and Metasploit to make sure our environment is securely hardened against vulnerabilities such as scanning with SQL injections. It is also important to make sure that our application is not vulnerable to CSS background attachments, and XSS (Cross Site Scripting) which enables attackers to inject client-side malicious scripts into web pages. An attacker can use it to bypass controls. We should enable alerts for items such as XSS, unsuccessful login attempts, to successful logins.

The site will use http session control logins to prevent non-authenticated users from entering the site. By creating a login, and logout account, the user must be authenticated to log into an account. If user does not have the proper credentials, they will not be able to log in. The http code will block any unauthenticated users from entering site. Once the page closes the user is automatically logged out. By looking for database syntax errors and adding unit tests we could ensure that certain types of uncontrolled user input would not be allowed in our database queries (p. 687.8).

By using an algorithm to transform the data to cipher text we can encrypt the content of our database, therefore rendering the text useless to attackers. Unless they can decipher the data, there is nothing they can use. There are multiple techniques and technologies we could use such as Transparent/External encryption (encrypts the entire database). Transparent involves encrypting “data at rest,” which is inactive data that is not being edited or pushed across the network. Also, symmetric, and asymmetric database encryption. They both involve a private key being applied to the data to call it from the database.

Lastly, our database must use TLS/SSL certificate from letsencrypt.org for production site. This site allows us to encrypt for free, and collaborate. The process of setting up our database, ensuring that it is a working and viable source is important. All the aspects and duties lead us to take a good look at how we can secure information not only in our homes, but in our work, our schools, and everywhere.

**Bibliography**

Database encryption. (2016, December 31). Retrieved March 01, 2017, from https://en.wikipedia.org/wiki/Database\_encryptionDatabase encryption. (2016, December 31). Retrieved March 01, 2017, from https://en.wikipedia.org/wiki/Database\_encryption

Google. (n.d.). Retrieved March 01, 2017, from https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=what is xss attack&\*

Kim, G., Debois, P., Willis, J., Humble, J., & Allspaw, J. (2016). The DevOps handbook: how to create world-class agility, reliability, and security in technology organizations. Portland, OR: IT Revolution Press, LLC.

Password Manager. (n.d.). Retrieved March 01, 2017, from https://identitysafe.norton.com/

Servlet HttpSession Login and Logout Example - javatpoint. (n.d.). Retrieved March 01, 2017, from http://www.javatpoint.com/servlet-http-session-login-and-logout-example

Username Generator. (n.d.). Retrieved March 01, 2017, from

20 Feb 2015 at 10:31, John Leyden tweet\_btn(). (n.d.). Evil CSS injection bug warning: Don't let hackers cross paths with your website. Retrieved March 01, 2017, from https://www.theregister.co.uk/2015/02/20/prssi\_web\_vuln/