Webware Authentication Project

Noah Olson - noahvolson - nvolson

This project completed for additional grad credit for CS 4241 Webware

Glitch Link: https://noah-olson-webauth.glitch.me/
Repository: https://github.com/noahvolson/WebAuthentication

Project Goal: Build a secure login system with a hashed database of passwords and encrypted user data.

Security Features

Encrypted requests: When using regular HTTP requests, the contents of POST messages can be read during transit by packet sniffing programs such as Wireshark. To avoid this, I enforce the use of HTTPS with custom middleware (see credit). All HTTP requests are now immediately redirected to the HTTPS version of my website.

Request headers: I use the express middleware “helmet” to apply several security-enhancing headers to further secure my HTTP responses. This mitigates a multitude of popular attacks: everything from clickjacking to cross-site scripting.

Rate limiting: I use the middleware “express-rate-limit” to limit the number of login and account creation attempts each user can make per day. There are three main benefits of this: limiting the potential for denial-of-service attacks, minimizing user enumeration via the “username taken” message, and making brute-force account cracking infeasible.

Hashed passwords: In the case of a plaintext database breach, malicious actors would have access to all user passwords. Every user and their corresponding data would be compromised. To avoid this, all passwords stored by my server are first hashed with bcrypt (which uses the blowfish hashing algorithm).

Password strength enforcement: By enforcing the complexity of passwords submitted to my database, I expand the dictionary necessary for the completion of brute-force attacks.

Encrypted user data: On account creation, each user inputs a secret message. Because this data is sensitive, it is important to avoid storing it in plaintext. Consequently all secret messages are encrypted using AES before they are stored in the database. Upon successful login, this message is decrypted and shown to the user.

Sessions: In order to maintain the login status of users across pages and tabs, I use the express-session middleware. On the client side, this stores only a signed session id. To improve the security of these generated cookies, I specify a generic cookie name to overwrite the default and minimize targeting. Additionally, I use the HttpOnly option to ensure that cookies are sent only via HTTP and not client side JavaScript. This helps defend against cross-site scripting.

Environment variables: My database password, session secret, and encryption key are all stored in environment variables instead of source code. This means that, even if my source code is leaked (it is available on glitch after all) user data will not be compromised.

Visual Design

While the design challenges of this project were minimal, I put care into ensuring a clean aesthetic.

I implemented a dark-theme for my login page. Such themes help to minimize eye strain and power draw. I adhered to materialize standards by ensuring that elements which should have a higher elevation are represented with lighter colors. One example of this is my buttons. On mouse over, their color becomes slightly darker to indicate that they are set into the page.

Challenges

Initially, I thought that JWT (Json Web Token) based authentication would be the best option for maintaining login state across tabs. The more I read up on it, the more I learned that this solution would not be appropriate for my problem: JWTs are largely used to authenticate APIs, not for authenticating web pages. There are many reasons for this. Most notably, tokens cannot be revoked in the same way that sessions can. If a user is abusing the system, tokens will grant them access until they expire, regardless of any input from administrators. Additionally, JWTs take up a considerable amount of space when compared to their counterparts. It is clear that maintaining server sessions is the cleanest solution for web page authentication.

Credit

Code snippet by Glitch user khalby786 used to force HTTPS
Link: https://support.glitch.com/t/tutorial-how-to-force-https/16669

Reasoning against JWTs for web page sessions
http://cryto.net/~joepie91/blog/2016/06/13/stop-using-jwt-for-sessions/

Lighter colors help increase elevation, darkmode scheme
https://material.io/archive/guidelines/style/color.html#color-themes

Reasons for dark mode
https://blog.weekdone.com/why-you-should-switch-on-dark-mode/