THE UNIVERSITY of ADELAIDE

Web and Database Computing

adelaide.edu.au

Web Security: Securing Your Web Application

Recap: Website login & Security

- Can be diffcult to implement well.
  - Very easy to make security mistakes.
- Risk of compromise
  - If compromised users may abandon your service or worse...

The 1st Rule of Security Programming

Don’t implement your own security

But if you have to, follow best practices!

Basics

1) Encrypt all communications

Use HTTPS for all communications involving sensitive information.

- Unencrypted HTTP traffic can be 'sniffed' on insecure networks, or read by third parties.
- HTTPS uses end-to-end encryption to ensure **confidentiality** between the client and server.
- This is especially important for any sites that have any form of login.
  - Not doing so exposes passwords and/or session tokens.
- Users accessing your website via HTTP should be redirected to the secure, HTTPS version of the site.
  - Do this for all requests, or use a HTTPS proxy like the one used on CS50.

2) Use Security Zones

Split your site and routes into separate zones that ensure only **authorized** users may access that content.

- Use middleware to enforce user permissions across an entire zone;
  - Avoid manually coding this check into every route. You are more likely to make mistakes when doing so.
- This also allows better enforcing of **authentication**, **confidentiality**, and allows us to **audit**/log differently for different zones.

3) Validate all inputs and outputs

Do not directly store user input without first checking to ensure it does not contain malicious content.

- Do not store objects sent from client side directly
    - Yes, I know that's what we've done so far.
- Use prepared statements to sanitize input data that will be used in database queries.
- express-validator has several functions for validating and escaping user input
    - https://github.com/chriso/validator.js
- When modifying webpage contents, use `innerText` instead of `innerHTML` wherever possible

Handling User Authentication

4) Get someone else to do it

Use OpenID where possible.

5) Don’t restrict passwords

Some web application programmers restrict character types and lengths to make sanitization easier.

- That's lazy.
- Don't be lazy!
- Allow all characters

Set long password max-lengths

- OWASP recommends 160 characters as a good length.

6) Don’t store passwords in plain text

Use a strong cryptographic hashing algorithm

- Argon2
- PBKDF2
- scrypt
- bcrypt
- Never use known compromised algorithms like MD5 or SHA-1

Cryptographic Hashing

A cryptographic hash function is a hash function which takes an input (or 'message') and returns a fxed-size alphanumeric string.

The ideal hash function has three main properties:

- It is extremely easy to calculate a hash for any given data. → Reduce load
- It is extremely computationally diffcult to calculate an alphanumeric text that has a given hash → Near impossible to reverse
- It is extremely unlikely that two slightly different messages will have the same hash → Near impossible to guess

7) Salt your hashes

Many hashing algorithms have been dumped into reverse-lookup tables known as rainbow tables, so a hash by itself is not secure.

Salting means:

- Use additional unique information when hashing a password.
- Store that unique information with the hash.

Now, if someone does fnd a match for the hash, it's not the original password.

Be sure your salt is:

- Fixed-length
- Crytographically Generated
- Random
- Different for each stored hash

8) Assume security will be compromised

Don't expect your current secure password system to protect your users forever:

- Keep your system up-to-date.
- Enforce good security practice on your users.
- Implement 2FA
- Plan for how you will handle a data breach.

General Advice & Summary

- Write Secure Code
  - Follow the best practices in OWASP's Guide to Building Secure Web Applications https://www.owasp.org/index.php/Guide
  - And the cheat sheets: https://cheatsheetseries.owasp.org/cheatsheets/Index.html
  - Use OWASP's Application Security Verification Standard as a guide to what an application needs to be secure https://www.owasp.org/index.php/ASVS
  - Use standard security components that are a fit for your organization
  - Use OWASP's ESAPI as a basis for your standard components https://www.owasp.org/index.php/ESAPI
- Review Your Applications
  - Have an expert team review your applications
  - Review your applications yourselves following OWASP Guidelines
  - OWASP Code Review Guide https://www.owasp.org/index.php/Code_Review_Guide
  - OWASP Testing Guide https://www.owasp.org/index.php/Testing_Guide

THE UNIVERSITY of ADELAIDE