Rupayan Das, Shane Eising, Zachary Frederick, Bridget Homer

Dr. Williams

CIS 4930

3/23/20

Mini Project 1: Salt/Pepper Hash and Web Exploit Prevention

Exploits and Mitigations

1. SQL Injection
   - **Vulnerability**: The input for the username search feature is not filtered.
   - **Initiation:** An attacker can enter an apostrophe to close the field, followed by SQL code such as "or 1=1--", causing the SQL query to return the entire database of usernames.
   - **Mitigation:** A SQL injection filter is placed on the input field, so search requests are sanitized and malicious SQL can't run.

2. IDOR/URL Manipulation
   - **Vulnerability:** "Tweet" content and the corresponding user's ID are sent as URL parameters.
   - **Initiation:** By editing the URL, attackers can alter the message and the user ID to change how tweets appear their dashboard.
   - **Mitigation:** Parameters are sent as post parameters, not URL parameters, so they are never exposed as plaintext.

3. CSRF/Session Attacks
   - **Vulnerability:** After logging in, plaintext session data is stored in a cookie.
   - **Initiation:** An attacker can manually change the "logged_in" cookie value to "true" or the "user_id" cookie value to any number via a browser developer console or by using software, such as Burp, to alter and replay a login request. This allows attackers to log in as anyone they want, without a password.
   - **Mitigation:** The session data is saved as a hash of specific attributes about the user's computer and browser, not as plaintext, making it much more difficult to forge.

4. XSS
   - **Vulnerability:** Users can set their profile pictures by entering an unfiltered website link.
   - **Initiation:** Instead of an image, an attacker could set their image link to code such as "'onerror = alert('xss_exploit')". When someone attempts to load this user's profile picture, the code will run.
   - **Mitigation:** Filenames are filtered; if a user does not link to a valid image, their profile picture is set to a default value.

Password Hashing vs. Encrypting

Hashing and encrypting passwords are two ways to obfuscate passwords in the database and in transit to the server. Encryption is a powerful tool, as passwords encrypted with long, random keys are very difficult to brute force and crack; however, if an attacker can access the database *and* the decryption key, the encryption is worthless. So, if the key is stored in the same place as the encrypted passwords or is easily cracked, password encryption is an extremely weak security choice.

Conversely, hashing is irreversible, meaning attackers can never convert a hashed password to plaintext. Despite this, hashed passwords are generally easier to guess than encrypted passwords. Knowing the hashing algorithm used, an attacker can run through a dictionary of common passwords, either by hashing every guess (brute force) or by using a rainbow table, a precomputed table of possible hashes. Still, the irreversible nature of hashing prevents a database from being completely compromised; there is no single key to reverse every password in the database and uncommon passwords are less likely to be in dictionaries and are therefore less likely to be cracked.

Salt and Pepper

Hashed passwords can also be secured with password salts and/or peppers. A password is "salted" when a salt – a random string – is concatenated to the password before it is hashed. When mounting an attack against salted passwords, the attacker must correctly guess the password *and* the salt. However, if the salt or salts are stored in the same database as the hashed passwords, the salts are no longer helpful if an attacker accesses the database.

To protect against this scenario, salted passwords can also have a pepper, which is essentially a second salt. A pepper is a single random string that is stored separately from the salts and hashes – for example, hardcoded into the website's source code. The pepper is added to all passwords before they're hashed, just like a salt, but if an attacker accesses the database – seeing all hashes and salts – they still need to guess or find the pepper. While peppers generally add more security, most common hashing algorithms are intended to be used with salts and adding a pepper may restrict the available algorithms and create unnecessary complications.