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

Team: Andrew Sanchez, Elijah Springer, Tyler Styers, Ying Xu

Our website simulates a bank web interface where users can register an account, log in to an account, view account information (e.g. funds available), search for existing users, and transfer funds to another account holder.

Attack 1

Vulnerability: SQL Injection in Search Form Input

Exploit: Logged-in users can search for existing users by knowing their username. Attackers can thus insert malicious input that will always evaluate to true into the search box. For example, inputting *hacker' OR '1=1* into the search box would yield all users, allowing the attacker to gain access to everyone's email and funds. Such information possesses a threat to user privacy and can be used to stage other attacks.

Mitigation: Sanitized user input by forbidding invalid characters that can lead to SQL statements, which prevents the injection from happening.

Attack 2

Vulnerability: IDOR in Login Form

Exploit: When logging in to the unprotected version of the site, the user's login email is sent as plain text in the POST request. The attacker can just intercept the request and edit the email to gain access to any user's bank account.

Mitigation: Used the JSON Web Token for a secure transfer of information rather than plain text email. The secure version would instead contain an access token in the GET request to the */home* page. In this way, attackers cannot perform IDOR by changing the email value since it is no longer visible anymore.

Attack 3

Vulnerability: Cookie-Based CSRF

Exploit: Steal the user's cookies by just accessing *document.cookie*, which contains the JSON Web Token (JWT), and then use the JWT to perform actions as the logged-in user.

Mitigation: Removed the cookies so that attackers cannot use this information to session hijack. Similar to mitigation for XSS.

Attack 4

Vulnerability: Cookie-Based XSS

Exploit: Attackers can steal the user's cookies via executing JavaScript in the search form. A possible input is **. With the access token given through the cookies, the attacker can use the JWT to perform actions as the logged-in user.

Mitigation: Removed the ability to execute JavaScript in the search form through input sanitization. Additionally, we moved away from using cookies, sending the access token in the request header instead, preventing cookie theft in this manner.

Password Hashing VS Encryption

Password hashing is a one-way transformation of passwords into a hash value, which is derived from the combination of both the password and a key using a set function, and cannot be reversed. In comparison, encryption is a two-way function. It uses an encryption algorithm to encode the password from plain text to cipher text. Passwords are scrambled but can be decrypted. Thus hackers who get access to a user database with encrypted passwords can easily decrypt them to find the original text. As such, hashing is a more secure method.

Salt and peppering passwords add another level of security to hashing. Both are random data appended to the password before the concatenated result goes into the hash function. They are helpful in preventing rainbow table attacks. The difference is that the salt value is often stored alongside the hash value while the pepper value is kept secret.