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

Initiation of Exploits, Types of Vulnerabilities, and Mitigations

The more complex and feature-rich a website is, the more vulnerabilities it may have and the bigger the attack surface may be. The types of vulnerabilities that can exist on web servers is enormous and always increasing, from Cross-Site Scripting (XSS) to XML External Entity (XXE) injection. Without the proper mitigations, vulnerabilities like these can lead to devastating effects on the server, including Denial of Service (DoS) and leaking of potentially sensitive data. Some of the most popular vulnerabilities are XSS, SQL Injection, Insecure Direct Object Reference (IDOR), and Cross-Site Request Forgery (CSRF).

XSS vulnerabilities occur when user input isn't properly escaped and is directly or indirectly conveyed back to the user or users. It's a type of code injection, specifically javascript, that, when successfully executed, can steal cookies and session tokens of other users, allowing an attacker to act as that user on the vulnerable website. Mitigating against XSS involves strictly filtering the user input and encoding output data to prevent it from being interpreted as any other language.

SQL injection is another common vulnerability in servers, specifically those that are using an SQL database to query site-based information, such as cookies, usernames, passwords, etc.. SQL injection occurs when user input isn't properly validated, allowing an attacker to, instead of entering in expected text, injects SQL code that is then executed by the server. User input is usually placed directly into the middle of an SQL statement, so an attacker can inject code that stops that query (such as a semicolon and/or comment) and then append a new query to be executed by the database, retrieving new, potentially personal data. Mitigating against SQL injection involves escaping user input, but especially using prepared statements, which sends the query and the actual input data to the server separately. Prepared statements are the best line of defense against SQL injection.

IDOR is a vulnerability that allows an attacker unauthorized access to a particular resource. IDOR involves any field, often a URL, that directly references an object to display to the user. In vulnerable systems, the server will not check the user's authentication when inputting into that field, meaning that an attacker can, with enough knowledge of the system (or an educated guess), reference objects that they should not have access to. To prevent against IDOR attacks, access control checks should be put in place to ensure only authorized users can access a requested object.

CSRF exists on a site due to a failure to properly set up same origin policy, which is used to prevent websites from interfering with each other. In this scenario, an attacker could set up a malicious site that, when navigated to, makes a request to a website that the victim is already logged into in another tab. This request will seem like it is coming from the logged-in victim, allowing the attacker to have the same privileges as that user. The harm this could do is it allows an attacker to potentially make bank account transfers or steal personal information. Mitigating against CSRF would be to make sure that the session cookie generation method is secure and can't be reversed, as well as making sure the generation occurs on the server side, not the client side.

Password Hashing and Encryption

Encryption on the other hand is a two way function, ie. it is reversible. We use an encryption key to encrypt the password and we can decrypt it if we have the key. It is unsafe as compared to hashing because the application usually has the key stored somewhere and if the attacker gets access to the key, he can decrypt all the encrypted passwords.

Password Hashing is defined as a one way function (or a mapping) which produces a hash value when a string is given. The worst that can happen is a collision, which means that 2 different plaintext values provide the same hashed value. Even though hashing is more secure than encryption, it can still be hacked: the attacker can pre hash every common password in a dictionary and do a dictionary attack. We add an extra layer of security on this by using the concept of Salting.

In the process of salting, we add a unique value to the end of the password to create a different hash value. This unique value is known as a salt. We can use a rainbow table where we add a salt before hashing our passwords. Since even a small change in the string produces an entirely different hash, it's more secure than the basic hashing technique.

The catch here is that usually the salt is stored as a plaintext in the database and it can still be used to hack into the passwords. The final concept we add on top of this is peppering. In peppering, we use a pepper as the key and the salted password as the input to make the final hash. Without the pepper, each and every password is completely secure.