Django Assignment 1

What is CSRF?

Cross-site request forgery (also known as CSRF) is a web security vulnerability that allows an attacker to induce users to perform actions that they do not intend to perform. It allows an attacker to partly circumvent the same origin policy, which is designed to prevent different websites from interfering with each other.

What is the impact of a CSRF attack?

In a successful CSRF attack, the attacker causes the victim user to carry out an action unintentionally. For example, this might be to change the email address on their account, to change their password, or to make a funds transfer. Depending on the nature of the action, the attacker might be able to gain full control over the user's account. If the compromised user has a privileged role within the application, then the attacker might be able to take full control of all the application's data and functionality.

How does CSRF work?

For a CSRF attack to be possible, three key conditions must be in place:

- **A relevant action.** There is an action within the application that the attacker has a reason to induce. This might be a privileged action (such as modifying permissions for other users) or any action on user-specific data (such as changing the user's own password).
- **Cookie-based session handling.** Performing the action involves issuing one or more HTTP requests, and the application relies solely on session cookies to identify the user who has made the requests. There is no other mechanism in place for tracking sessions or validating user requests.
- **No unpredictable request parameters.** The requests that perform the action do not contain any parameters whose values the attacker cannot determine or guess. For example, when causing a user to change their password, the function is not vulnerable if an attacker needs to know the value of the existing password.

What are CSRF tokens?

A CSRF token is a unique, secret, unpredictable value that is generated by the server-side application and transmitted to the client in such a way that it is included in a subsequent HTTP request made by the client. When the later request is made, the server-side application validates that the request includes the expected token and rejects the request if the token is missing or invalid.

CSRF tokens can prevent CSRF attacks by making it impossible for an attacker to construct a fully valid HTTP request suitable for feeding to a victim user. Since the

attacker cannot determine or predict the value of a user's CSRF token, they cannot construct a request with all the parameters that are necessary for the application to honor the request.

How should CSRF tokens be generated?

CSRF tokens should contain significant entropy and be strongly unpredictable, with the same properties as session tokens in general.

You should use a cryptographic strength pseudo-random number generator (PRNG), seeded with the timestamp when it was created plus a static secret.

If you need further assurance beyond the strength of the PRNG, you can generate individual tokens by concatenating its output with some user-specific entropy and take a strong hash of the whole structure. This presents an additional barrier to an attacker who attempts to analyze the tokens based on a sample that are issued to them

How should CSRF tokens be validated?

When a CSRF token is generated, it should be stored server-side within the user's session data. When a subsequent request is received that requires validation, the server-side application should verify that the request includes a token which matches the value that was stored in the user's session. This validation must be performed regardless of the HTTP method or content type of the request. If the request does not contain any token at all, it should be rejected in the same way as when an invalid token is present.