Secure WordPress Development

Guidelines for Custom Plugin & Theme Coding Standards

1. Core Principles of Secure Development

The baseline security strategy for any custom WordPress plugin or theme development is: **Never trust user input, always sanitize on entry, and always escape on output.**

2. Input Sanitization

Sanitization removes dangerous or unwanted characters from user inputs before the data is processed or saved to the database.

```
sanitize_text_field( $_POST['user_input'] ); // Clean basic text input
sanitize_email( $_POST['email'] );      // Sanitize email string
esc_url_raw( $_POST['website'] );       // Safe URL sanitization for DB entry
```

3. Output Escaping

Escaping strips out malicious scripts and prevents Cross-Site Scripting (XSS) attacks when displaying database or user-supplied content to the client.

```
echo esc_html( $user_name );       // Escape general text inside HTML tags
echo esc_attr( $field_value );     // Escape text inside HTML attributes
echo esc_url( $redirect_link );    // Escape URLs before printing in href
echo wp_kses_post( $custom_html ); // Sanitize HTML preserving default allowed tags
```

4. Nonce Validation (CSRF Prevention)

Nonces (Number Used Once) ensure that requests are initiated by authorized users and protect forms and AJAX actions from Cross-Site Request Forgery.

```
// Generate Nonce
wp_nonce_field( 'save_profile_action', 'profile_nonce' );

// Verify Nonce on Submission
if ( ! isset( $_POST['profile_nonce'] ) || ! wp_verify_nonce( $_POST['profile_nonce'], 'sav
e_profile_action' ) ) {
    wp_die( 'Security check failed.' );
}
```

5. Safe Database Queries (SQLi Prevention)

Direct SQL execution is highly vulnerable. Always use prepared statements via $wpdb->prepare() to prevent SQL Injection attacks.

```
global $wpdb;
// Safe Query Preparation
$query = $wpdb->prepare(
    "SELECT * FROM {$wpdb->prefix}custom_table WHERE status = %s AND user_id = %d",
    $status_string,
    $user_id_integer
);
$results = $wpdb->get_results( $query );
```