Common Vulnerabilities

- Types of Vulnerabilities
  - SQL Injection
  - Cross Site Scripting (XSS)
  - Cross-site Request Forgery (CSRF)
  - Staying Current

Security is an ever-changing landscape, and vulnerabilities evolve over time. The following is a discussion of common vulnerabilities you should protect against, and the techniques for protecting your theme from exploitation.

Types of Vulnerabilities

SQL Injection

SQL injection happens when values being inputted are not properly sanitized allowing for any SQL commands in the inputted data to potentially be executed. To prevent this, the WordPress API is extensive, offering functions like `add_post_meta();` instead of you needing to adding the post meta manually via SQL (`INSERT INTO wp_postmeta…`).

> exploits_of_a_mom

xkcd Exploits of a Mom

The first rule for hardening your theme against SQL injection is: When there's a WordPress function, use it.

But sometimes you need to do complex queries, that have not been accounted for in the API. If this is the case, always use the `$wpdb` functions. These were built specifically to protect your database.

All data in SQL queries must be SQL-escaped before the SQL query is executed to prevent against SQL injection attacks. The best function to use for SQL-escaping is `$wpdb->prepare()` which supports both a sprintf()-like and vsprintf()-like syntax.

```
$wpdb->get_var( $wpdb->prepare(
    "SELECT something FROM table WHERE foo = %s and status = %d",
    $name, // an unescaped string (function will do the sanitization for you)
    $status // an untrusted integer (function will do the sanitization for you)
) );
```

Cross Site Scripting (XSS)

Cross Site Scripting (XSS) happens when a nefarious party injects JavaScript into a web page.

Avoid XSS vulnerabilities by escaping output, stripping out unwanted data. As a theme's primary responsibility is outputting content, a theme should escape dynamic content with the proper function depending on the type of the content.

An example of one of the escaping functions is escaping URL from a user profile.

```
<img src="<?php echo esc_url( $great_user_picture_url ); ?>" />
```

Content that has HTML entities within can be sanitized to allow only specified HTML elements.

```
$allowed_html = array(
    'a' => array(
        'href' => array(),
        'title' => array()
    ),
    'br' => array(),
    'em' => array(),
    'strong' => array(),
);

echo wp_kses( $custom_content, $allowed_html );
```

Cross-site Request Forgery (CSRF)

Cross-site request forgery or CSRF (pronounced sea-surf) is when a nefarious party tricks a user into performing an unwanted action within a web application they are authenticated in. For example, a phishing email might contain a link to a page that would delete a user's account in the WordPress admin.

If your theme includes any HTML or HTTP-based form submissions, use a nonce to guarantee a user intends to perform an action.

```
<form method="post">
    <!-- some inputs here … -->
    <?php wp_nonce_field( 'name_of_my_action', 'name_of_nonce_field' ); ?>
</form>
```

Staying Current

It is important to stay current on potential security holes. The following resources provide a good starting point:

- WordPress Security Whitepaper
- WordPress Security Release
- Open Web Application Security Project (OWASP) Top 10

□□□ #2
□ □□□ □□□ 2026-10-31 18:08:05 UTC
□ □□□ □□□ 2026-10-31 18:08:06 UTC