
Static Application Security Testing (SAST)

Our development environment utilizes **PHP CodeSniffer (PHPCS)** configured with the **WordPress Coding Standards (WordPress-Extra ruleset)** and **PHP Compatibility standards** to identify potential security and quality issues early in the development lifecycle.

- **Implementation:**
  Before any code is committed to the repository, it is scanned to ensure strict adherence to WordPress naming conventions, proper input sanitization, secure database escaping, and overall coding best practices.
- **Code Verification:**
  The codebase (e.g., `KCTZoomController.php`) reflects the results of these scans through:
  - Consistent use of strict typing (e.g., `: WP_REST_Response`)
  - Namespace encapsulation (e.g., `namespace KCTApp\controllers\api;`)
  - Mandatory `permission_callback` validation on all REST API routes to enforce authorization controls

---

Dynamic Application Security Testing (DAST)

We conduct dynamic security testing using **OWASP ZAP (Zed Attack Proxy)** against our **staging environment** to identify runtime vulnerabilities.

- **Scope:**
  Dynamic testing focuses on critical endpoints, including:
  - Zoom Authorization endpoint (`/kivicare/v1/settings/zoom-telemed/authorize`)
  - Webhook listener endpoint (`/webhook`)
- These endpoints are tested to ensure secure handling of:
  - Invalid or malformed payloads
  - Missing or invalid signatures
  - Replay attack attempts
- Testing confirms that the application does not expose system errors, sensitive configuration details, or access tokens under adverse conditions.
