Securing PHP CodeIgniter Forms Against Spam and Vulnerabilities

To secure your forms in a PHP CodeIgniter codebase and prevent spam or unauthorized data insertion, you can implement the following measures:

1. **Implement CSRF Protection**
   - Enable CSRF protection in `application/config/config.php`:
     ```php
     $config['csrf_protection'] = TRUE;
     $config['csrf_token_name'] = 'csrf_token';
     $config['csrf_cookie_name'] = 'csrf_cookie';
     ```
     - This prevents malicious users from submitting forms via third-party websites.

2. **Validate and Sanitize Input**
   - Use CodeIgniter's form validation library:
     ```php
     $this->form_validation->set_rules('email', 'Email', 'required|valid_email');
     $this->form_validation->set_rules('name', 'Name', 'required|alpha');
     ```
   - Sanitize inputs:
     ```php
     $name = $this->input->post('name', TRUE);
     ```

3. **Use reCAPTCHA**

   - Integrate Google reCAPTCHA:

     ```php
     <script src="https://www.google.com/recaptcha/api.js" async defer></script>
     <div class="g-recaptcha" data-sitekey="your-site-key"></div>
     ```

   - Verify server-side:

     ```php
     $captchaResponse = $this->input->post('g-recaptcha-response');
     ```

4. **Limit Form Submission Rate**

   - Store submission timestamps and restrict rapid submissions.

5. **Honeypot Fields**

   - Add hidden fields:

     ```php
     <input type="text" name="honeypot" style="display:none">
     ```

   - Reject submissions with filled honeypots.

6. **Validate Referrer Headers**

   - Check `HTTP_REFERER` to ensure requests come from your domain:

     ```php
     if (strpos($_SERVER['HTTP_REFERER'], 'yourdomain.com') === false) { }
     ```

7. **Use CAPTCHA for Suspicious Behavior**

- Dynamically show CAPTCHA for repeated submissions.

8. **Log and Monitor Activity**

- Log form submissions (IP, timestamp, user agent) and identify patterns.

9. **Restrict Allowed Input Types**

- Use appropriate HTML input types (`type="email"`, `type="number"`) and validate server-side.

10. **Email Verification**

- Send confirmation emails for critical forms before processing data.

11. **Disable Unused Fields**

- Remove unused fields and avoid processing unnecessary inputs.

12. **Enable Database Constraints**

- Use unique indexes, foreign keys, and strict column types.

13. **Rate Limit or Block Suspicious IPs**

- Use tools like Cloudflare or WAF to block suspicious IPs or rate-limit requests.

14. **Update CodeIgniter and PHP**

- Regularly update frameworks and libraries to avoid vulnerabilities.

Summary of Steps to Implement:

1. Enable CSRF protection in CodeIgniter.

2. Use form validation and sanitization.

3. Integrate reCAPTCHA for public-facing forms.

4. Add honeypot fields.

5. Implement rate-limiting and IP monitoring.

6. Log all activity and monitor patterns.

By combining these measures, your forms will be significantly more secure and resilient against spam attacks.