**🔹 Problem Statement (From the Slide)**

You need to **develop a script to analyze PHP and JavaScript source code** for insecure coding practices, such as:

* Weak cryptographic algorithms
* Unsanitized inputs
* Unsafe/dangerous function usage
* Poor error handling

The goal is **early vulnerability detection** and **secure software development**.

**🔹 What the Project Should Do**

Your project should be a **static code analyzer tool** (like a lightweight version of SonarQube, ESLint, or Bandit) that:

1. **Scans PHP and JavaScript code** for insecure patterns.
2. **Detects vulnerabilities** such as:
   * SQL Injection (unsanitized user input in DB queries).
   * XSS (user input directly injected into HTML).
   * Use of dangerous functions (eval, exec, system, etc.).
   * Weak cryptographic algorithms (MD5, SHA1, DES).
   * Poor error handling (printing stack traces, exposing sensitive info).
3. **Outputs developer-friendly reports** in JSON format (and optionally CLI output).
4. **Provides remediation suggestions** (not just flagging errors, but telling developers how to fix them).
5. **Ranks vulnerabilities by severity** (High, Medium, Low).

**🔹 Solution Approach (From the Slide, Expanded)**

✅ Use **pattern matching + AST parsing**:

* Parse the code into an **Abstract Syntax Tree (AST)** (using libraries like tree-sitter, esprima, php-parser).
* Traverse the syntax tree to identify risky coding constructs.

✅ Vulnerability detection logic:

* **Regex/Patterns**: for spotting functions like eval(), md5().
* **AST Rules**: to detect unsanitized input flows ($\_GET, $\_POST in PHP, or req.query in Node.js without validation).

✅ Annotate vulnerabilities:

* Attach **explanations + suggestions** (e.g., “Avoid md5(); use bcrypt or argon2”).

✅ Generate output:

* Developer-friendly JSON reports.
* Command-line summaries for quick feedback.

**🔹 Key Criteria (Hackathon Winning Edge)**

1. **Accuracy** – Minimize false positives (don’t just flag everything).
2. **Integration** – Provide a CLI tool that can be plugged into GitHub Actions, Jenkins, or CI/CD pipelines.
3. **Usability** – Reports should be easy to read and actionable.
4. **Compliance** – Follow **OWASP Top 10** guidelines for web vulnerabilities.

**🔹 Technology Stacks (From the Slide)**

* **Languages**: Python (main analyzer), Node.js (optional for JS parsing).
* **Parsing Libraries**:
  + esprima (JS parser),
  + tree-sitter (multi-language parser),
  + php-parser for PHP.
* **Output**: JSON reports, CLI summaries.

**🔹 Outcomes**

* A **command-line static code analyzer**.
* Detects **critical vulnerabilities** in PHP & JavaScript.
* Provides **remediation guidance + severity levels**.
* Output reports in **JSON format** (usable in CI/CD).

**🔹 What You Need to Do to Make It Hackathon-Winning 🚀**

1. **MVP in First Few Hours**
   * Implement a basic scanner that detects a few vulnerabilities (like eval() or md5()).
   * Show JSON/CLI output.
2. **Expand with Smart Rules**
   * Add SQL Injection detection (look for string concatenation in DB queries).
   * Add XSS detection (user input written into HTML).
3. **Severity Ranking System**
   * High: eval(), SQL Injection.
   * Medium: md5(), SHA1.
   * Low: Bad error handling (console.log(err) or die($e) in PHP).
4. **User-Friendly Reports**
   * JSON reports + command-line summary.
   * Example:
   * {
   * "file": "app.js",
   * "line": 23,
   * "issue": "Use of eval() detected",
   * "severity": "HIGH",
   * "suggestion": "Avoid eval(); use JSON.parse() or safe alternatives"
   * }
5. **Extra Hackathon Edge Features**
   * **GitHub Action integration** → so developers can see security issues in pull requests.
   * **Visualization dashboard (bonus)** → simple web UI that shows vulnerabilities and severity stats.
   * **Remediation tips** → link to OWASP or official docs for fixing vulnerabilities.
   * **Extensible design** → allow adding more rules in YAML/JSON config file.

✅ **In short**: Your hackathon project should be a **static analyzer CLI tool** for PHP and JS, detecting vulnerabilities (unsanitized inputs, unsafe functions, weak crypto, poor error handling), reporting them with **severity levels & fixes**, and generating **JSON reports** that can be used in CI/CD pipelines.

That combination of **accuracy, usability, OWASP compliance, and CI/CD integration** is what will make your project stand out as hackathon-winning.