Chall1.php

* **总漏洞数**: 2

1. **漏洞类型**: 不安全的反序列化 (Insecure Deserialization)
   * **CWE 标号**: CWE-502
   * **描述**: 使用 unserialize() 反序列化来自用户输入的数据，存在不安全的反序列化风险。攻击者可以通过构造恶意数据执行任意代码。
   * **修复建议**: 避免直接反序列化用户输入的数据，使用更安全的数据格式（如 JSON），并进行严格的输入验证。
2. **漏洞类型**: 文件删除 (Unintended File Deletion)
   * **CWE 标号**: CWE-20
   * **描述**: 在对象销毁时删除文件，如果 cache\_file 属性被恶意设置，可能导致删除任意文件。
   * **修复建议**: 对要删除的文件路径进行严格验证，确保只能删除特定目录下的文件。

Chall2.php

* **总漏洞数**: 1

1. **漏洞类型**: 不安全的反序列化 (Insecure Deserialization)
   * **CWE 标号**: CWE-502
   * **描述**: 使用 unserialize() 反序列化来自用户输入（如 Cookie）的数据，存在不安全的反序列化风险。攻击者可以通过构造恶意数据执行任意代码。尤其是 \_\_wakeup() 方法中使用 eval() 执行反序列化数据，风险更高。
   * **修复建议**: 避免直接反序列化用户输入的数据，使用更安全的数据格式（如 JSON），并进行严格的输入验证。避免使用 eval()，改用更安全的代码执行方式。

Tarlogic1.php

* **总漏洞数**: 1

1. **漏洞类型**: 不安全的反序列化 (Insecure Deserialization)
   * **CWE 标号**: CWE-502
   * **描述**: 使用 unserialize() 反序列化来自用户输入的数据，存在不安全的反序列化风险。攻击者可以通过构造恶意数据执行任意命令，特别是在 \_\_wakeup() 方法中使用 system() 执行命令。
   * **修复建议**: 避免直接反序列化用户输入的数据，使用更安全的数据格式（如 JSON），并进行严格的输入验证。避免在对象的魔术方法中执行命令。

Tarlogic-ex1.php

* **总漏洞数**: 1

1. **漏洞类型**: 不安全的反序列化 (Insecure Deserialization)
   * **CWE 标号**: CWE-502
   * **描述**: 使用 unserialize() 反序列化来自用户输入的数据，存在不安全的反序列化风险。攻击者可以通过构造恶意数据执行任意代码，或者绕过验证逻辑。
   * **修复建议**: 避免直接反序列化用户输入的数据，使用更安全的数据格式（如 JSON），并进行严格的输入验证。

tarlogic—ex2.php

* **总漏洞数**: 1

1. **漏洞类型**: 不安全的反序列化 (Insecure Deserialization)
   * **CWE 标号**: CWE-502
   * **描述**: 使用 unserialize() 反序列化来自用户输入的数据，存在不安全的反序列化风险。攻击者可以通过构造恶意数据执行任意代码，特别是结合类的 \_\_destruct() 方法来获取敏感信息，如 flag。
   * **修复建议**: 避免直接反序列化用户输入的数据，使用更安全的数据格式（如 JSON），并进行严格的输入验证。避免在析构方法中执行敏感操作。